Marco Tomamichel

EE5139/EE6139: Information Theory and its Applications

(Semester I, 2021–2022)

Disclaimer: These notes are not yet free of typos or always presented in the most clear way. Any comments that help reduce these deficiencies are very much appreciated. Parts of Chapters 0 and Chapter 7 are based on notes by Vincent Y. F. Tan. Most figures and the section on LDPC codes were contributed by Michael X. Cao.

Contents

$\emptyset$	empty set $\{\}$
$[M]$	the set $\{1, 2, \ldots, M\}$
$\mathcal{P}(\mathcal{X})$	the power set of $\mathcal{X}$, i.e. $\{A : A \subseteq \mathcal{X}\}$
$\mathcal{X} \times \mathcal{Y}$	the set of tuples $\{(x, y) : x \in \mathcal{X}, y \in \mathcal{Y}\}$
$\mathcal{X}^n$	the set of n-tuples with each element taking values in $\mathcal{X}$, e.g., $\mathcal{X}^2 = \mathcal{X} \times \mathcal{X}$
$\{0,1\}^n$	the set of n-bit strings
$\{0,1\}^*$	the set of bit strings of arbitrary length
$\max \mathcal{X}$	largest $x^* \in \mathcal{X}$, might not always exist
$\sup \mathcal{X}$	smallest $x^* \in \mathbb{R}$ such that $x \leq x^*$ for all $x \in \mathcal{X}$;
	equals the maximum, $\max \mathcal{X}$, if it exists
$\min \mathcal{X}$	smallest $x^* \in \mathcal{X}$, might not always exist
$\inf \mathcal{X}$	largest $x^* \in \mathbb{R}$ such that $x \geq x^*$ for all $x \in \mathcal{X}$;
	equals the minimum, $\min \mathcal{X}$, if it exists
$\mathbf{1}\{x = y\}$	indicator function, evaluates to 1 if the condition is true and 0 otherwise,
	so that, for example, $\mathbf{1}\{x = y\} + \mathbf{1}\{x \neq y\} = 1$
δ_{xy}	shorthand for $\mathbf{1}\{x = y\}$
$P_X(x)$	probability mass function (pmf), $P_X(x) = P[X = x]$
$p_X(x)$	probability density function (pdf), i.e. $P[X \in (1,2)] = \int_1^2 p_X(x)\mathrm{d}x$
$P[X \in \mathcal{A}]$	probability of a random variable X being in some set $\mathcal{A}$,
	i.e. $P[X \in \mathcal{A}] = \mathbb{P}(\{\omega : X(\omega) \in \mathcal{A}\}) = \sum_{x \in \mathcal{A}} P_X(x)$
$P[5 \leq X < 6]$	another way of writing $P[X \in [5, 6)]$
$\log$	logarithm; in these notes we take the logarithm to base 2, i.e. $\log = \log_2$

Table 1: Some basic notation used in this module.

pmf	probability mass function
pdf	probability density function
cdf	cumulative density function
rv	random variable
DMS	discrete memoryless source
DMC	discrete memoryless channel

Table 2: Some abbreviations used in this module.

Chapter 0

Review of mathematical notation and foundations

[Week 1]

Intended learning outcomes:

- You are familiar with common notation used throughout the lecture.
- You are comfortable with the main mathematical concepts needed in this module, namely basic probability theory including random variables, conditional probabilities and Markov chains.
- You can apply basic bounds on tail probabilities, and can prove the weak law of large numbers.
- You can compute vector norms and apply the Cauchy-Schwarz inequality.
- You know what convex and concave functions are and can apply Jensen's inequality.
- You know what finite fields are and how to come up with the multiplication table for simple examples.

0.1 Notation

We will use standard notation and abbreviations that you should be familiar with from other modules. Some of the less frequently encountered mathematical expressions are summarised in Tables 1 and 2 on the previous page.

0.2 Probability theory

We will not directly need the framework of probability theory in its most abstract formulation as presented in the following, but it is good to know that both discrete and continuous random variables can be seen as emanating from a shared mathematical framework.

0.2.1 Probability space

A probability space is represented by a triple $(\Omega, \Sigma, \mathbb{P})$. Here Ω is a set that is called the *sample space*. Moreover, Σ is a σ-algebra, i.e. a collection of subsets of Ω, called events, with the following properties:

- $\Omega \in \Sigma$
- If $A \in \Sigma$, then its *complement*, $A^c = \Omega \setminus A$ is also in Σ, i.e. $A^c \in \Sigma$.
- If $A_1, A_2, \ldots, A_n, \ldots \in \Sigma$, then $\bigcup_{i=1}^{\infty} A_i \in \Sigma$

Question 0.1. *Show that the above also implies that $\emptyset \in \Sigma$ and $\bigcap_{i=1}^{\infty} A_i \in \Sigma$.*

For example, let $\Omega = [0, 1]$, and we are interested in the probability of subsets of Ω that are intervals of the form $[a, b]$ where $0 \leq a < b \leq 1$, but not individual points in Ω. Then we should also be able to say something about the probability of the union, intersections, complement and so on of such intervals. This is captured by the definition of a σ-algebra. Think of Σ as the properties of Ω that can actually be observed.

Example 0.2. *If your random variable is the location an athlete lands after a long jump then it makes sense to take Ω to be positive real numbers, $\mathbb{R}_+$ indicating the distance jumped (say, in meters). However, even with arbitrarily good equipment we cannot actually measure a real number, we can only ever say that he landed in some interval, the size of which is given by our measurement precision. Thus, Σ, comprised of the events we can actually observe, is built up by including all (arbitrarily small) intervals in $\mathbb{R}_+$ and their unions and complements. Or another way of looking at this is that the probability of the jumper landing exactly at $9m$ is always zero — it is simply the wrong question to ask. But the probability of landing within $1cm$ or some arbitrarily small interval around $9m$ might very well be nonzero.*

Question 0.3. *For the advanced reader: Note however that in the above example $\{x\} \in \Sigma$ for any $x \in \mathbb{R}^+$, that is, single points are also elements of the σ-algebra. Can you see why? Use an infinite intersection to construct it.*

Finally, the probability measure $\mathbb{P}$ is a function $\mathbb{P} : \Sigma \to [0, 1]$ defined on the measurable space (Ω, Σ), and represents your "belief" about the events in Σ. In order for $\mathbb{P}$ to be called a probability measure, it must satisfy the following two properties:

1. $\mathbb{P}(\Omega) = 1$

2. For $A_1, A_2, \ldots$ such that $A_i \cap A_j = \emptyset$ for all $i \neq j$, i.e. for mutually *disjoint* sets, we have

$$\mathbb{P}\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mathbb{P}(A_i) \tag{1}$$

Some basic and very useful properties that can be derived from the above definition. The union bound in particular is very often used when analysing problems in information theory.

Proposition 0.4. *Let $(\Omega, \Sigma, \mathbb{P})$ be a probability space. The following holds true:*

1. $\mathbb{P}(A^c) = 1 - \mathbb{P}(A)$

2. If $A \subset B$, then $\mathbb{P}(A) \leq \mathbb{P}(B)$

3. $\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B) \leq \mathbb{P}(A) + \mathbb{P}(B)$, *which is called the* union bound. *Clearly, by induction, the union bound works for finitely many sets $A_i, i = 1, \ldots, k$, namely*

$$\mathbb{P}\left(\bigcup_{i=1}^{\infty} A_i\right) \leq \sum_{i=1}^{\infty} \mathbb{P}(A_i). \tag{2}$$

Proof. Property 1 follows since $A^c \cap A = \emptyset$, and thus $\mathbb{P}(A) + \mathbb{P}(A^c) = \mathbb{P}(\Omega) = 1$ by (1). For Property 2, note that $B \setminus A = B \cap A^c \in \Sigma$ and since $A \cap (B \setminus A) = \emptyset$ we again argue that $\mathbb{P}(A) + \mathbb{P}(B \setminus A) = \mathbb{P}(B)$, from which the desired inequality follows.

For Property 3 note that $A \cup B$ can be decomposed in three different ways into mutually disjoint sets:

$$A \cup B = A \cup (B \setminus A) = B \cup (A \setminus B) = (A \setminus B) \cup (B \setminus A) \cup (A \cap B)\,. \tag{3}$$

Again using (1) for each of these decompositions we have

$$\begin{align} 2\mathbb{P}(A \cup B) &= \mathbb{P}(A) + \mathbb{P}(B) + \mathbb{P}(B \setminus A) + \mathbb{P}(A \setminus B) \tag{4} \\ &= \mathbb{P}(A) + \mathbb{P}(B) + \mathbb{P}(A \cup B) - \mathbb{P}(A \cap B)\,, \tag{5} \end{align}$$

which implies the desired equality. □

Question 0.5. *Show that $0 \leq \mathbb{P}(A) \leq 1$ for every $A \in \Sigma$.*

Sometimes we have two conflicting beliefs, or models, about the underlying probability distribution, and so we will consider two compatible probability spaces $(\Omega, \Sigma, \mathbb{P})$ and $(\Omega, \Sigma, \mathbb{Q})$. They offer different predictions about the probability with which the events in Σ occur, and one fundamental task in statistics is to find out which model is the correct one from the frequency with which certain events occur. We will cover this later in the module.

0.2.2 Random variables

We will usually not deal directly with the probability space but with random variables. A *random variable* (rv) $X : \Omega \to \mathcal{X}$ is a function from the space (Ω, Σ) to a measurable space $(\mathcal{X}, \Sigma_X)$. In order for X to make any sense, the mapping has to ensure that $\{\omega \in \Omega : X(\omega) \in \mathcal{B}\} \in \Sigma$ for all $\mathcal{B} \in \Sigma_X$, because we are restricted to observing events in Σ and our random variable can thus not be more fine-grained than what Σ allows. Functions satisfying this property are called a *measurable function.* A random variable is then more formally defined as a measurable mapping from (Ω, Σ) to $(\mathcal{X}, \Sigma_X)$.

The only two examples of interest for us in the following are discrete and continuous random variables:

discrete rv: $\mathcal{X}$ is a discrete set and Σ_X is the power set $\mathcal{P}(\mathcal{X})$ of $\mathcal{X}$, i.e. the set of all subsets of $\mathcal{X}$.

continuous rv: $\mathcal{X} = \mathbb{R}$ and $\Sigma_X = \mathcal{B}$, the Borel σ-algebra. This is the smallest σ-algebra containing all open intervals in $\mathbb{R}$.

The probability measure $\mathbb{P}$ induces a probability measure P_X on $(\mathcal{X}, \Sigma_X)$, given by

$$P_X(B) = P[X \in B] = \mathbb{P}(\{\omega \in \Omega : X(\omega) \in B\}) \tag{6}$$

for all $B \in \Sigma'$. P_X is called the *distribution* of the random variable X.

If $\mathcal{X} = \{a_1, \ldots, a_d\}$ is discrete (and Σ_X the power set of $\mathcal{X}$), then we say that X is a *discrete random variable.* The distribution of X is then also known as the *probability mass function (pmf)* of X and is fully characterised by all the events consisting of a single value, i.e. the values $P_X(a_1), P_X(a_2), \ldots, P_X(a_d)$.

Question 0.6. *Can you give a formal argument why the values at these points are sufficient?*

Some random variables are not random at all. If there is an a_i with $P_X(a_i) = 1$ (and thus $P_X(a_j) = 0$ for all $j \neq i$), then we call this random variable *deterministic.* On the other extreme we have *uniformly distributed* random variables, where $P_X(a_i) = \frac{1}{d}$ for all $i \in [d]$.

Example 0.7. *The simplest example is the Bernoulli random variable. It is defined on a binary alphabet* $\mathcal{X} = \{0, 1\}$ *and we write* $X \sim \mathrm{Bern}(\epsilon)$ *to denote the rv with* $P[X = 1] = \epsilon$ *and* $P[X = 0] = 1 - \epsilon$.

Let us now consider a real-valued random variable X. If there exists a function $p_X : \mathbb{R} \to [0, \infty)$ such that for all $A \in \Sigma_X$, we have

$$P[X \in A] = \int_A p_X(x)\,\mathrm{d}x \tag{7}$$

then we say that X is a *continuous random variable.* The function p_X is called the *probability density function (pdf)* of X. We also define the *cumulative distribution function* (cdf) by integrating $p_X(x)$, that is, the cdf is given by $F_X(a) = \mathbb{P}[X \leq a] = \int_{-\infty}^{a} p_X(x)\,\mathrm{d}x$.

Question 0.8. *Show that $\int_{\mathcal{X}} p_X(x) = 1$. Moreover, if p_X is continuous at some point x, it must satisfy $p_X(x) \geq 0$. Can $p_X(x)$ ever be larger than 1?*

In this class, we deal mainly with discrete rvs, although we will also encounter Gaussian random variables, which are continuous, later on.

Example 0.9. *We denote the pdf of a* Gaussian *random variable X as*

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{8}$$

where μ is the mean *and σ the* standard deviation *of X. The* variance *of X is σ^2. A* normal Gaussian *random variable has $\mu = 0$ and $\sigma = 1$. The corresponding cdf is denoted as*

$$\Phi(y) = \int_{-\infty}^{y} \mathcal{N}(x; 0, 1)\, \mathrm{d}x. \tag{9}$$

Some additional notations and definitions for discrete random variables are given below. The counterparts for continuous random variables can be obtained by simply replacing pmfs with pdfs. Thus assume now that X and Y are discrete random variables taking on values in $\mathcal{X}$ and $\mathcal{Y}$ respectively. The joint pmf of X and Y is defined as

$$P_{X,Y}(x, y) = P[X = x \wedge Y = y] = \mathbb{P}\big(\{\omega \in \Omega : X(\omega) = x \wedge Y(\omega) = y\}\big)\,. \tag{10}$$

Question 0.10. *Verify that $P_Y(y) = \sum_{x' \in \mathcal{X}} P_{X,Y}(x', y)$.*

With this in hand we can define conditional pmf's and a notion of independence of random variables.

- The conditional pmf is given by

$$P_{X|Y}(x|y) = \frac{P_{X,Y}(x, y)}{P_Y(y)} = \frac{P_{Y|X}(y|x) P_X(x)}{P_Y(y)}, \quad \text{for} \quad P_Y(y) > 0\,, \tag{11}$$

  where the second expression is often referred to as Bayes' rule. If $P_Y(y) = 0$ then the conditional pmf is simply not defined.

- X and Y are *independent random variables*, if and only if, for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$,

$$P_{X,Y}(x, y) = P_X(x) P_Y(y) \tag{12}$$

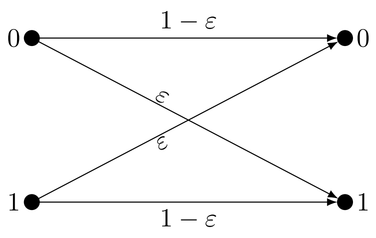
  or equivalently $P_{X|Y}(x|y) = P_X(x)$. The latter condition simply states that the conditional distribution $P_{X|Y}(x|y)$ does not depend on y.

Example 0.11 (Binary symmetric channel)**.** *$X \sim$ Bern(p) is a bit that is sent over channel and is corrupted by additive noise $Z \sim$ Bern(ϵ), where X and Z are independent. The output of the channel is $Y = X \oplus Z$. The channel is fully defined by the conditional distribution $P_{Y|X}$, which we can compute as follows:*

$$P_{Y|X}(y|x) = P[X \oplus Z = y \mid X = x] = P[Z = y \oplus x \mid X = x] = P[Z = y \oplus x] = P_Z(y \oplus x) \tag{13}$$

Hence, the channel can be given as a matrix or pictorially as follows:

x	y	$P_{Y\|X}$
0	*0*	$1-\epsilon$
1	*0*	ϵ
0	*1*	ϵ
1	*1*	$1-\epsilon$



0.2.3 Expectation and variance

The expectation of a random variable X is defined to be

$$\mathbb{E}[X] = \int_\Omega X(\omega)\,\mathrm{d}\mathbb{P}(\omega). \tag{14}$$

This definition has a very precise mathematical meaning in measure theory, but here we are only interested in two special cases. If X is a discrete random variable this reduces to the familiar formula

$$\mathbb{E}[X] = \sum_{x\in\mathcal{X}} xP_X(x)\,. \tag{15}$$

If X is a continuous random variable with pdf $f_X(x)$, we have

$$\mathbb{E}[X] = \int_\mathbb{R} xp_X(x)\,\mathrm{d}x\,. \tag{16}$$

Note that the expectation is a statistical summary of the distribution of X, rather than depending on the realised value of X. If there are two different models $\mathbb{P}$ and $\mathbb{Q}$ we need to specify which probability measure we are using. We only do this when necessary (because the the model is not obvious from context) by adding a subscript $\mathbb{E}_P$ or $\mathbb{E}_Q$.

If g is a function, the expectation of $g(X)$ is given by

$$\mathbb{E}[g(X)] = \int_\mathbb{R} g(x)p_X(x)\,\mathrm{d}x. \tag{17}$$

Question 0.12. *Show that the expectation is linear, i.e.* $\mathbb{E}[aX+b] = a\mathbb{E}[X]+b$.

The variance of X is the expectation of $g(X) = (X-\mathbb{E}[X])^2$. Thus,

$$\mathrm{Var}(X) = \mathbb{E}[(X-\mathbb{E}[X])^2] = \int_\mathbb{R}(x-\mathbb{E}[X])^2 p_X(x)\,\mathrm{d}x\,. \tag{18}$$

Question 0.13. *Check from the above definition that the variance can also be expressed as*

$$\mathrm{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2. \tag{19}$$

Question 0.14. *Verify that* $\mathcal{N}(x;\mu,\sigma^2)$ *indeed has expectation* μ *and variance* σ^2.

0.2.4 Markov chains

Markov chains describe a notion of conditional independence. Let's start with the three random variables X, Y and Z. They are said to form a *Markov chain in the order*

$$X - Y - Z$$

if their joint distribution P_{XYZ} satisfies

$$P_{XYZ}(x,y,z) = P_X(x)P_{Y|X}(y|x)P_{Z|Y}(z|y) \qquad \text{for all} \qquad (x,y,z) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}. \tag{20}$$

This the same as saying that X and Z are *conditionally independent given* Y.

Question 0.15. *Assume $X - Y - Z$. Show that it is also true that $Z - Y - X$.*

Notice that if we do not assume anything about the joint distribution P_{XYZ}, then it factorizes (by repeated applications of Bayes rule) as

$$P_{XYZ}(x,y,z) = P_X(x)P_{Y|X}(y|x)P_{Z|XY}(z|x,y) \qquad \text{for all} \qquad (x,y,z) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \tag{21}$$

so what Markovianty in the order $X - Y - Z$ buys us is that $P_{Z|XY}(z|x,y) = P_{Z|Y}(z|y)$ (i.e., we can drop the conditioning on X). In essence all the information that we can learn about Z is already contained in Y. No other information about Z can be gleaned from knowing X if we already know Y. Another way of saying this is that the conditional distribution of X and Z given $Y = y$ can be factorised as

$$P_{XZ|Y}(x,z|y) = P_{X|Y}(x|y)P_{Z|Y}(z|y) \qquad \text{for all} \qquad (x,y,z) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}. \tag{22}$$

Notice that this is in direct analogy to the situation where X and Z are (marginally) independent. Simply set Y to be a deterministic random variable (with only one possible outcome) to recover the definition of independence.

Question 0.16. *If Z is a deterministic function of Y, show that $X - Y - Z$ is true.*

Question 0.17. *If X and Z are conditionally independent given Y, this does not imply that X and Z are marginally independent (in general). Construct a counterexample.*

0.3 Tail bounds

In this section, we summarise some bounds on probabilities that we use extensively in the sequel. More precisely, we are interested in showing that the probability of a random variable deviating too far from its expectation value is small.

0.3.1 Basic bounds

We start with the familiar Markov and Chebyshev inequalities.

Proposition 0.18 (Markov's inequality)**.** *Let X be a real-valued non-negative random variable with pdf p_X. Then for any $a > 0$, we have*

$$P[X > a] \leq \frac{\mathbb{E}[X]}{a}. \tag{23}$$

Proof. By the definition of the expectation, we have

$$\mathbb{E}[X] = \int_0^\infty x p_X(x)\,\mathrm{d}x \geq \int_a^\infty x p_X(x)\,\mathrm{d}x \geq a \int_a^\infty p_X(x)\,\mathrm{d}x = aP[X > a]. \tag{24}$$

and we are done. □

Note that this bound only becomes nontrivial if a exceeds the expectation value $\mathbb{E}[X]$.

Question 0.19. *In which step is non-negativity of X used?*

Question 0.20. *Can you do the proof also for discrete random variables?*

If we let X above be the non-negative random variable $(X - \mathbb{E}[X])^2$, we obtain Chebyshev's inequality.

Proposition 0.21 (Chebyshev's inequality)**.** *Let X be a real-valued random variable with mean μ and variance σ^2. Then for any $a > 0$, we have*

$$P\big[|X - \mu| > a\sigma\big] \leq \frac{1}{a^2}. \tag{25}$$

Proof. Let X in Markov's inequality be the random variable $g(X) = (X - \mathbb{E}[X])^2$. This is clearly non-negative and the expectation of $g(X)$ is $\mathrm{Var}(X) = \sigma^2$. Thus, by Markov's inequality, we have

$$P[g(X) > a^2\sigma^2] \leq \frac{\sigma^2}{a^2\sigma^2} = \frac{1}{a^2}. \tag{26}$$

Now, $g(X) > a^2\sigma^2$ if and only if $|X - \mu| > a\sigma$ so the claim is proved. □

We now consider a collection of real-valued random variables that are independent and identically distributed (i.i.d.). In particular, let $X^n = (X_1, \ldots, X_n)$ be a collection of independent random variables where each X_i has distribution P with zero mean and finite variance σ^2.

Proposition 0.22 (Weak Law of Large Numbers)**.** *For every $\epsilon > 0$, we have*

$$\lim_{n\to\infty} P\left[\left|\frac{1}{n}\sum_{i=1}^n X_i\right| > \epsilon\right] = 0. \tag{27}$$

Consequently, the average $\frac{1}{n}\sum_{i=1}^n X_i$ converges to 0 in probability.

Note that for a sequence of random variables $\{S_n\}_{n=1}^{\infty}$, we say that this sequence *converges to a number $b \in \mathbb{R}$ in probability* if for all $\epsilon > 0$,

$$\lim_{n\to\infty} P\big[|S_n - b| > \epsilon\big] = 0. \tag{28}$$

We also write this as $S_n \xrightarrow{\text{p}} b$. Contrast this to convergence of numbers: We say that a sequence of numbers $\{s_n\}_{n=1}^{\infty}$ *converges to a number $b \in \mathbb{R}$* if we have $\lim_{n\to\infty} |s_n - b| = 0$.

Proof. Let $\frac{1}{n}\sum_{i=1}^{n} X_i$ take the role of X in Chebyshev's inequality. Clearly, the mean is zero. The variance of X is

$$\text{Var}\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{1}{n^2}\text{Var}\left(\sum_{i=1}^{n} X_i\right) = \frac{1}{n^2}\sum_{i=1}^{n}\text{Var}(X_i) = \frac{\sigma^2}{n}. \tag{29}$$

Thus, we have

$$P\left(\left|\frac{1}{n}\sum_{i=1}^{n} X_i\right| > \epsilon\right) \leq \frac{\sigma^2}{n\epsilon^2} \longrightarrow 0 \tag{30}$$

as $n \to \infty$, which proves the claim. □

Some further useful bounds are derived in the homework.

0.3.2 Central limit theorem

We can actually say quite a bit more than the weak law of large numbers dictates. If the scaling in front of the sum in the statement of the law of large numbers Proposition 0.22 is $1/\sqrt{n}$ instead of $1/n$, the resultant random variable $\frac{1}{\sqrt{n}}\sum_{i=1}^{n} X_i$ converges in distribution to a Gaussian random variable. As in Proposition 0.22, let X^n be a collection of i.i.d. random variables where each X_i is zero mean with finite variance σ^2.

Proposition 0.23 (Central limit theorem)**.** *For any $a \in \mathbb{R}$, we have*

$$\lim_{n\to\infty} P\left(\frac{1}{\sigma\sqrt{n}}\sum_{i=1}^{n} X_i < a\right) = \Phi(a). \tag{31}$$

In other words,

$$\frac{1}{\sigma\sqrt{n}}\sum_{i=1}^{n} X_i \xrightarrow{\text{d}} Z \tag{32}$$

where $\xrightarrow{\text{d}}$ means convergence in distribution and Z is a standard Gaussian random variable.

For a sequence of random variables $\{S_n\}_{n=1}^{\infty}$, we say that this sequence of random variables *converges in distribution* to another random variable $\bar{S}$ if

$$\lim_{n\to\infty} P(S_n < a) = P(\bar{S} < a)$$

for all $a \in \mathbb{R}$. The proof of this statement requires tools that are outside the scope of these notes, but can be found in any textbook on probability theory.

0.4 Vector norms and Cauchy-Schwarz inequality

We can naturally interpret pmf's on an alphabet with d symbols as row vectors in a d-dimensional inner-product space. Without loss of generality we take the alphabet to be $\mathcal{X} = \{1, 2, \dots, d\} = [d]$ and define the vector $p \in \mathbb{R}^d$ by its elements $p_x = P_X(x)$ for $x \in [d]$. The inner product is denoted by $\langle \cdot, \cdot \rangle$. For two general vectors $u, v \in \mathbb{R}^d$, it evaluates to

$$\langle u, v \rangle = uv^T = \sum_{i=1}^{d} u_i v_i \,, \tag{33}$$

where v^T denotes the transpose of the vector v, and is a column vector. The Cauchy-Schwarz inequality then states that for any two vectors we have

$$|\langle u, v \rangle|^2 \leq \langle u, u \rangle \langle v, v \rangle \,. \tag{34}$$

On these vector spaces we can also define the p-norms for $p \geq 1$ as

$$\|u\|_p = \left(\sum_{x=1}^{d} |u_x|^p \right)^{\frac{1}{p}} \tag{35}$$

We will mostly encounter the 1-norm and the 2-norm, the latter being the usual Euclidian norm of the vector. The following special case of the Cauchy-Schwarz inequality will be encountered later.

Lemma 0.24. *Let $u, v \in \mathbb{R}^d$. Then,*

$$\|u \cdot v\|_1 \leq \|u\|_2\| \, \|v\|_2 \,, \tag{36}$$

where $\cdot$ denotes the element-wise product of the vectors, i.e. $(u \cdot v)_i = u_i v_i$.

Proof. Define $k \in$ using $k_i = \operatorname{sgn}^*(u_i v_i)$, where sgn^* is the modified sign function, i.e.

$$\operatorname{sgn}^*(x) = \begin{cases} -1 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \,. \tag{37}$$

Then since $\operatorname{sgn}^*(x)^2 = 1$ for all $x \in \mathbb{R}$, the Cauchy-Schwarz inequality yields

$$\big|\langle k \cdot u, v \rangle\big| \leq \langle u, u \rangle \langle v, v \rangle = \|k \cdot u\|_2 \|v\|_2 = \|u\|_2 \|v\|_2 \,. \tag{38}$$

Moreover, we have

$$\langle k \cdot u, v \rangle = \sum_{x=1}^{d} k_i u_i v_i = \sum_{x=1}^{d} |u_i v_i| = \|u \cdot v\|_1 \,. \tag{39}$$

□

Question 0.25. *Using the above, can you show that $\|u\|_1 \leq \sqrt{d}\|u\|_2$?*

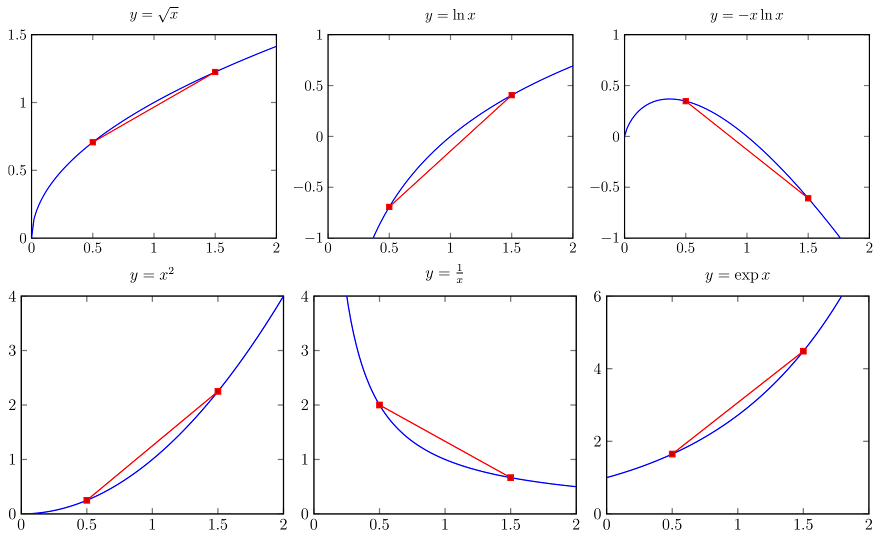


Figure 0.1: Examples of concave (upper ones) and convex (lower ones) functions. The straight line between two points of the curve is either below or above the plot of the function, which is exactly what the definition requires.

0.5 Convexity and Jensen's inequality

A function $f(x)$ is said to be *convex* on [a, b] if for all $x, y \in [a, b]$ and $\lambda \in [0, 1]$, we have

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y). \tag{40}$$

If we do not mention any interval then we mean that the function is convex on its full domain, i.e. the statement $\log(x)$ is concave should be understood as $\log(x)$ is concave on $(0, \infty)$.

The function f is *strictly convex* if equality in (40) holds only if $\lambda = 0$ or 1, or $x = y$. The function f is *concave* if $-f$ is convex, and *strictly concave* if $-f$ is strictly convex.

In the homework you will show the following lemma:

Lemma 0.26. *If f is convex on $[a, b]$, then for any $a \leq x_1 < x_2 \leq x_3 < x_4 \leq b$, we have*

$$\frac{f(x_2) - f(x_1)}{x_2 - x_1} \leq \frac{f(x_4) - f(x_3)}{x_4 - x_3} \tag{41}$$

Proposition 0.27 (Jensen's inequality)**.** *If $f(x)$ is convex and X is a random variable on $\mathbb{R}$, then*

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)] \tag{42}$$

We only give a proof for discrete distributions here.

Proof. We give a proof by induction. Due to convexity, we have

$$p_1 f(x_1) + p_2 f(x_2) \geq f(p_1 x_1 + p_2 x_2), \tag{43}$$

which proves the statement if $|\mathcal{X}| = 2$.

Suppose the statement $\mathbb{E}[f(X)] \geq f(\mathbb{E}[X])$ is true when $|\mathcal{X}| = k-1$. Then consider a pmf with k mass points $\{p_1, p_2, \ldots, p_k\}$. Define another pmf on $k-1$ points given by the probabilities

$$p_i' = \frac{p_i}{1-p_k}, \quad i = 1, \ldots, k-1\,. \tag{44}$$

We then have

$$\begin{aligned}\sum_{i=1}^{k} p_i f(x_i) &= p_k f(x_k) + (1-p_k)\sum_{i=1}^{k-1} p_i' f(x_i) && (45)\\ &\geq p_k f(x_k) + (1-p_k) f\left(\sum_{i=1}^{k-1} p_i' x_i\right) && (46)\\ &\geq f\left(p_k x_k + (1-p_k)\sum_{i=1}^{k-1} p_i' x_i\right) && (47)\\ &= f\left(\sum_{i=1}^{k} p_i x_i\right) && (48)\end{aligned}$$

where the first inequality is from the induction hypothesis and the second by convexity (of two points). By the definition of expectation we have $\mathbb{E}[f(X)] \geq f(\mathbb{E}[X])$. □

Often it is hard to check convexity directly. But for twice differentiable functions, this is easy.

Proposition 0.28. *Let $f : [a,b] \to \mathbb{R}$ be twice differentiable. The function f is convex if and only if $f''(x) \geq 0$ for all $x \in (a,b)$, and strictly convex if $f''(x) > 0$ for all $x \in (a,b)$.*

Proof. Assume $f''(x) > 0$ for all $x \in [a,b]$. By Taylor expansion of f around $x_0 \in (a,b)$, we have

$$f(x) = f(x_0) + f'(x_0)(x-x_0) + \frac{f''(x^*)}{2}(x-x_0)^2 \tag{49}$$

where $x^* \in [x_0, x]$. By assumption $f''(x^*) > 0$ so the quadratic term is strictly positive unless $x = x_0$, in which case it is still non-negative. Now let $x_0 = \lambda x_1 + (1-\lambda)x_2$. Further let $x = x_1$. Then we have

$$f(x_1) \geq f(x_0) + f'(x_0)((1-\lambda)(x_1 - x_2)). \tag{50}$$

Now let $x = x_2$. Then we have

$$f(x_2) \geq f(x_0) + f'(x_0)(\lambda(x_2 - x_1)). \tag{51}$$

Both of these inequalities are strict unless $\lambda \in \{0,\}]$ or $x_1 = x_2$. Multiplying the first inequality by λ and the second by $1-\lambda$ and adding them up, we recover the definition of strict convexity. If we instead had assumed only $f''(x) \geq 0$ the same argument would ensure convexity (but no longer strict convexity).

For the other direction, choose $a < x_1 < x_2 < x_3 < x_4 < b$. By the property shown in Lemma 0.26,

$$\frac{f(x_2) - f(x_1)}{x_2 - x_1} \leq \frac{f(x_4) - f(x_3)}{x_4 - x_3} \tag{52}$$

Now let $x_2 \searrow x_1$ and $x_3 \nearrow x_4$. We see that $f'(x_1) \leq f'(x_4)$, and since these were arbitrary points, f' is increasing on (a, b). So $f''(x) \geq 0$ for all $x \in (a, b)$. □

0.6 Finite field arithmetic

This is a rather informal discussion, but it is sufficient for our purposes.

A finite field is a field (on which addition, subtraction, multiplication and division are defined) with a finite number of elements. Such fields are denoted by F_q where q is the number of elements in the field, or its *dimension*. For each dimension, the field is unique up to a relabelling of the elements. The idea is that such fields behave like $\mathbb{Q}$, $\mathbb{R}$ or $\mathbb{C}$, with the usual rules for addition and multiplication.

A bit more formally, we have two binary operations on F_q denoted by $+$ and $\cdot$ and the following properties (here a, b and c are any elements of F_q):

Associativity: $a + (b + c) = (a + b) + c$ and $a \cdot (b \cdot c) = (a \cdot b) \cdot c$.

Commutativity: $a + b = b + a$ and $a \cdot b = b \cdot a$.

Identities: There exist two different elements $0, 1$ such that $a + 0 = a$ and $a \cdot 1 = a$.

Additive inverse: Every a has an additive inverse, denoted $-a$, such that $a + (-a) = 0$.

Multiplicative inverse: Every $a \neq 0$ has a multiplicative inverse, denoted a^{-1}, such that $a \cdot a^{-1} = 1$.

Distributivity: $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$.

Such fields exist only for particular numbers of elements, namely when $q = p^\ell$ for some prime p and $\ell \in \mathbb{N}$.

For F_q where q is prime we can always simply denote the elements of F_q by the integers $\{0, 1, \ldots, q-1\}$ and use integer addition and multiplication modulo q as our operations.

Question 0.29. *Verify the above properties for F_2 and F_3. Can you find the inverse of 2 for a general F_q with prime q? Use that $q+1$ is even...*

This strategy fails when $q = 4$ is not a prime. The problem is that using multiplication modulo 4 we would, for example, get

$$\begin{aligned} 2 \times 0 &= 0 && (53)\\ 2 \times 1 &= 2 && (54)\\ 2 \times 2 &= 4 \mod 4 = 0 && (55)\\ 2 \times 3 &= 6 \mod 4 = 2\,, && (56) \end{aligned}$$

and hence 2 does not have a multiplicative inverse.

When q is a prime power $q = p^\ell$ we can derive the arithmetic using a polynomial ring. We give the construction here; but we do not attempt to show that it actually works or that it is unique. First, we denote the elements of F_q by strings of length ℓ with elements in F_p. In particular, if the underlying prime is 2, these are simply binary strings, e.g., $F_4 = \{00, 01, 10, 11\}$. We can then interpret these elements as polynomials of degree $\ell - 1$ with coefficients in F_p. Again, for F_4 the polynomials corresponding to the four elements would be $00 \to 0$, $01 \to 1$, $10 \to x$ and $11 \to x+1$. We can add these polynomials modulo p for each coefficient individually, so in particular for the binary case the negation of each number is just the number itself. For multiplication, we simply do this modulo an irreducible polynomial. (An irreducible polynomial is one that has no roots in F_p.) The choice of irreducible polynomial turns out not to matter — the resulting fields are equivalent up to relabelling of elements. For F_4 we can take the irreducible polynomial to be x^2+x+1.

Question 0.30. *Verify that $x^2 + x + 1$ is indeed irreducible over F_2? Is it also irreducible over F_3?*

So for the above labelings $\{00, 01, 10, 11\}$ of elements, we get

$$\begin{aligned} &10 \times 00 \to x \times 0 = 0 \to 00 && \implies 10 \times 00 = 00 && (57)\\ &10 \times 01 \to x \times 1 = x \to 10 && \implies 10 \times 01 = 10 && (58)\\ &10 \times 10 \to x \times x = x^2 \mod x^2+x+1 = x+1 \to 11 && \implies 10 \times 10 = 11 && (59)\\ &10 \times 11 \to x \times (x+1) = x^2 + x \mod x^2+x+1 = 1 \to 01 && \implies 10 \times 11 = 01\,. && (60) \end{aligned}$$

Hence, 10 and 11 are multiplicative inverses of each other. The full addition and multiplication tables can then be written down as follows:

+	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

×	00	01	10	11
00	00	00	00	00
01	00	01	10	11
10	00	10	11	01
11	00	11	01	10

Similar constructions can be done for every prime power, and, quite importantly for practical applications, all of this arithmetic can be implemented highly efficiently in computer programs.

Chapter 1

Information measures

[Week 2]

Intended learning outcomes:

- You can compute the entropy and conditional entropy for any discrete random variable and understand the basic properties of these two quantities, e.g. you can apply the chain rule or sub-additivity.
- You can compute mutual information and now how it relates to entropy and conditional entropy. You can apply the data-processing inequality for mutual information.
- You can compute the relative entropy and understand how entropy and mutual information can be expressed in terms of the relative entropy.

Book reference: Chapter 2 in Cover & Thomas [**?**], but we are not following it too closely.

1.1 Surprisal and entropy

It is not immediately clear how to model our intuitive notion of "information" in a mathematical language. In this chapter we take a somewhat axiomatic approach to information measures, i.e. we try to build them up from our intuitive understanding of what entropy and information "should" be. But we will only really be able to justify the choices we make here once we start analysing practical problems in information theory, and see that the quantities we derive here pop up again and again.

1.1.1 Surprisal

It turns out to be fruitful to start not by finding an expression for the information contained in a random variable, but rather the lack of information, or uncertainty inherent in a random experiment. Let us consider a discrete random variable X taking values in $\mathcal{X}$ following the pmf $P_X(x) = p_x$. How surprised are we to see a particular outcome $x \in \mathcal{X}$ of this random experiment? Clearly this depends on the probability p_x and not the value of x itself. In fact, we do not even need to know what $\mathcal{X}$ really is. On the one hand, if $p_x = 1$ we are not surprised at all since we already knew that we would see x. On the other hand, the smaller p_x is the more surprised we are to see this particular outcome. If $p_x = 0$ we will never see x, so our surprise when seeing it anyway would be literally off the scale. Furthermore — and this turns out to be a very convenient choice — if we do a random experiment twice independently and both times observe x, we say that we will be twice as surprised as if we had seen x once in a single random experiment.

The above notions can be formalised, and that is essentially what Shannon did when he introduce the notion of *surprisal.* Let us denote the surprisal of x as $s(p_x)$. We want this function to satisfy the following three conditions:

1. **Monotonicity:** $s(p_x) = 0$ if $p_x = 1$ and $s(p_x)$ increases monotonically as p_x decreases.

2. **Additivity:** The surprisal of seeing a pair of outcomes of independent random experiments is simply the sum of the individual surprisals, i.e. $s(p_x p_y) = s(p_x) + s(p_y)$.

3. **Normalisation:** $s(\frac{1}{2}) = 1$

Question 1.1. *We do not really need the condition $s(p_x) = 0$ if $p_x = 1$ under Point 1 as it follows directly from additivity. Can you see how?*

It turns out that the only positive function that satisfies these three properties is the logarithm. To show this one uses a result by Erdös that characterises additive functions, but that is beyond the scope here. We therefore pick

$$s(p_x) = \log \frac{1}{p_x} \,. \tag{1.1}$$

where the logarithm is taken to base 2 (as everywhere in these notes) so that the normalisation requirement is satisfied.

We can see the surprisal as another random variable, say S, that takes the value $s(p_x) = \log \frac{1}{p_x}$ with probability p_x. Since $S = S(X)$ is a function of X we usually simply write this new random variable as

$$S(X) = \log \frac{1}{P_X(X)} \,. \tag{1.2}$$

1.1.2 Entropy

Entropy measures how much we can learn by looking at the outcome of a random experiments, or, in other words, how much uncertainty there is about the outcome. It is simply the expected surprisal of X.

Definition 1.2. *Given a discrete random variable X, the* entropy *of X is defined as*

$$H(X) := \mathbb{E}[S(X)] = \mathbb{E}\left[\log \frac{1}{P_X(X)}\right] = \sum_{x \in \mathcal{X}} p_x \log \frac{1}{p_x} \tag{1.3}$$

Here and throughout we use the convention that $0 \log 0 = 0$. This is reasonable since $\lim_{\epsilon\to 0} \epsilon \log \epsilon = 0$, and thus we simply continuously extend the function to the point 0.

Question 1.3. *Can you verify $\epsilon \log \epsilon \to 0$ as $\epsilon \to 0$?*

Note again that the entropy X is really only a function of the pmf of X, and in particular independent of the alphabet $\mathcal{X}$, in contrast to potential alternative uncertainty measures like the variance of X.

Sometimes we are interested in more than just the expected surprisal. The minimum surprisal, or min-entropy, for example, has applications in cryptography (see Chapter 3) and the variance of $S(X)$ has itself operational meaning in many information-theoretic problems when we go beyond first order asymptotics.

Question 1.4. *Can you find an expression for $\operatorname{Var}[S(X)]$ in terms of the probabilities p_x?*

Now let us explore the entropy a bit. First we want to show the following basic property.

Proposition 1.5. *Let X be a discrete random variable taking values in $\mathcal{X}$. We have*

$$H(X) \geq 0, \tag{1.4}$$

with equality if and only if X is deterministic.

Proof. Since $p_x \leq 1$, we have $\log \frac{1}{p_x} \geq 0$ for every $x \in \mathcal{X}$, so the expectation of this quantity over x must be non-negative too. In fact, $\log \frac{1}{p_x}$ equals 0 if and only if $p_x = 1$ and hence $H(X) = 0$ only if there exists an $x \in \mathcal{X}$ for which $p_x = 1$, which is the hallmark of a deterministic rv. □

The entropy is a strictly concave function of the probability mass function P_X. To see this, we first verify that $f(t) = t \log \frac{1}{t} = -t \log t$ is concave on $(0, 1)$ by taking its second derivative:

$$f'(t) = -\log t - \log e, \qquad f''(t) = -\frac{\log e}{t}\,. \tag{1.5}$$

Since the latter is always negative for $t \in (0, 1)$, the function is indeed strictly concave according to Lemma 0.28. Now since the entropy is simply the sum $\sum_{x\in\mathcal{X}} f(p_x)$ it is indeed a concave function of the pmf. This simple property, together with Jensen's inequality, has profound implications. The first one is that the entropy has a unique maximum. Intuitively we would want that entropy is maximal when uncertainty about the outcome is greatest, namely when the rv is uniformly distributed. And this is indeed the case.

Proposition 1.6. *Let X be a discrete random variable taking values in $\mathcal{X}$. We have*

$$H(X) \leq \log |\mathcal{X}|, \tag{1.6}$$

with equality if and only if X is uniformly distributed.

The general case will be covered in the homework but here we give a proof for the case when the set $\mathcal{X}$ is a bit, i.e. when the random variable is binary.

Proof for $\mathcal{X} = \{0,1\}$. It is easy to verify by a simple computation that $H(X) = 1$ for a uniformly distributed random variable, so the difficulty is only in showing that this is the maximum and only achieved for the uniform distribution.

Let now $\{p, 1-p\}$ for $p \in [0,1]$ be a general pmf for the random variable X. We use the function $f(t) = -t \log t$ to simplify notation. Then we can write

$$\begin{aligned}
H(X) &= f(p) + f(1-p) && (1.7)\\
&= \frac{1}{2}\left(f(p) + f(1-p)\right) + \frac{1}{2}\left(f(p) + f(1-p)\right) && (1.8)\\
&\le f\left(\frac{1}{2}p + \frac{1}{2}(1-p)\right) + f\left(\frac{1}{2}p + \frac{1}{2}(1-p)\right) && (1.9)\\
&= f\left(\frac{1}{2}\right) + f\left(\frac{1}{2}\right) && (1.10)\\
&= 1\,. && (1.11)
\end{aligned}$$

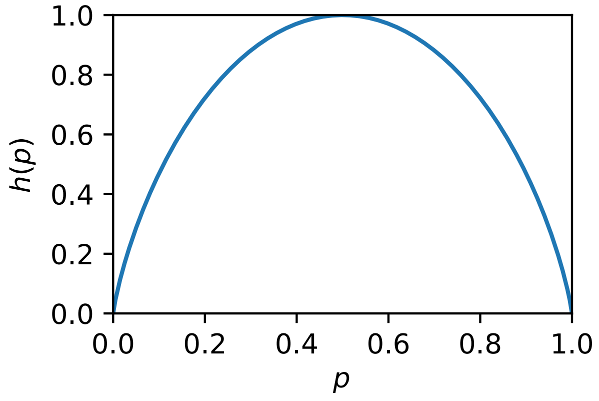
The inequality is due to Jensen's inequality and the strict concavity of f, and equality holds only if $p = 1 - p = \frac{1}{2}$, i.e. when the random variable X follows the uniform distribution. □

Concavity in fact has even stronger consequences, and we will show a few additional properties of entropy later on using it.

Example 1.7. *The simplest example of a random variable is the Bernoulli random variable X with $\mathcal{X} = \{0,1\}$ and $P_X(0) = p$ for $p \in [0,1]$. The entropy of the Bernoulli random variable is called the* binary entropy*,*

$$H(X) = p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p} =: h(p)\,. \qquad (1.12)$$

From the plot we can easily verify all the properties we discussed above.



1.2 Conditional entropy, and mutual information

1.2.1 Joint entropy

For two discrete random variables X and Y with joint pmf $P_{XY}(x, y) = p_{xy}$ we can simply consider (X, Y) as one single random variable and use the same construction to define the surprisal of a tuple (X, Y) as $S(X, Y) = -\log P_{XY}(X, Y)$. Its expectation is the *joint entropy*, $H(XY)$, given by

$$H(XY) := \mathbb{E}[S(X, Y)] = \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p_{xy} \log \frac{1}{p_{xy}} \tag{1.13}$$

The first thing to note is that — if X and Y are independent — then $p_{xy} = p_x \cdot p_y$ and thus the expression simplifies to

$$\begin{aligned} H(XY) &= \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p_{xy} \log \frac{1}{p_x p_y} && (1.14)\\ &= \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p_{xy} \log \frac{1}{p_x} + \log \frac{1}{p_y} && (1.15)\\ &= \sum_{x\in\mathcal{X}} p_x \log \frac{1}{p_x} + \sum_{y\in\mathcal{Y}} p_y \log \frac{1}{p_y} && (1.16)\\ &= H(X) + H(Y)\,. && (1.17) \end{aligned}$$

This is not true in general though if the two random variables are correlated.

Question 1.8. *Find an example for which $H(XY) = H(X) = H(Y) = 1$.*

1.2.2 Conditional entropy

So why do these entropies not just add up? Fundamentally, this is because once we learn X we might not be so surprised seeing some particular outcomes of the random variable Y anymore. In fact, in the most extreme case, we have $Y = f(X)$ for some function f; hence, once we know that X takes on the value x, we can immediately deduce that Y will take on the value $f(x)$ with probability one, and thus there is no surprisal anymore! We model this "conditional suprisal" using the conditional pmfs, $P_{Y|X}(y|x) = p_{y|x}$, which leads us to conditional entropy.

Definition 1.9. *The* conditional entropy *of Y given X is defined as*

$$H(Y|X) = \mathbb{E}\left[\log \frac{1}{P_{Y|X}(Y|X)}\right] = \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p_{xy} \log \frac{1}{p_{y|x}}\,. \tag{1.18}$$

This can be interpreted as the expectation of the entropy of Y over all outcomes X. We sometimes use the notation $H(Y|X=x) = H(Y_x)$ to denote the entropy of the random variable Y_x that follows the pmf $\{p_{y|x}\}_{y\in\mathcal{Y}}$, i.e., the pmf of Y when we already know that $X = x$. Using this and the expression in (1.18) we can write the conditional entropy as

$$H(Y|X) = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p_{xy}\log\frac{1}{p_{y|x}} \tag{1.19}$$

$$= \sum_{x\in\mathcal{X}} p_x \sum_{y\in\mathcal{Y}} p_{y|x}\log\frac{1}{p_{y|x}} \tag{1.20}$$

$$= \sum_{x} p_x\, H(Y|X=x)\,. \tag{1.21}$$

The last line which expresses the conditional entropy in terms of an average of (unconditional) entropies is particularly useful since it allows us to immediately conclude that the conditional entropy is also bounded from below and above, like the entropy. We thus have

$$0 \le H(Y|X) \le \log|\mathcal{Y}|\,. \tag{1.22}$$

Moreover, our definition of conditional entropy also allows us to establish a *chain rule* for the conditional entropy, which sometimes is in fact used as the definition of conditional entropy itself. This rule is very useful because it allows us to write the joint entropy as a sum of its parts, even if the two random variables are not independent.

Proposition 1.10. *We have $H(XY) = H(X) + H(Y|X)$.*

Proof. We take advantage of $p_{xy} = p_x p_{y|x}$ to write

$$H(XY) = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p_{xy}\log\frac{1}{p_{xy}} \tag{1.23}$$

$$= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p_{xy}\log\frac{1}{p_x} + \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p_{xy}\log\frac{1}{p_{y|x}} \tag{1.24}$$

$$= \sum_{x\in\mathcal{X}} p_x\log\frac{1}{p_x} + \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p_{xy}\log\frac{1}{p_{y|x}} \tag{1.25}$$

$$= H(X) + H(Y|X)\,. \tag{1.26}$$

□

Question 1.11. *Show that $H(Y|X) = H(Y)$ for independent random variables. Using the chain rule, find a different proof that $H(XY) = H(X) + H(Y)$ in this case.*

Now we have put everything in place to show our first entropic inequality, which relates the entropy of two random variables with their joint entropy. This result shows the *subadditivity* of entropy.

Proposition 1.12. *Let X and Y be two discrete random variables. Then*

$$H(XY) \leq H(X) + H(Y) \qquad \textit{or, equivalently,} \qquad H(X|Y) \leq H(X)\,. \tag{1.27}$$

Equality holds in either statement only if X and Y are independent.

Proof. The equivalence of the two relations follows directly from the chain rule, we thus only need to show the second statement.

We start with Eq. (1.21), which states that

$$\begin{aligned} H(Y|X) &= \sum_x p_x H(Y|X=x) && (1.28)\\ &= \sum_x p_x \sum_y p_{y|x} \log \frac{1}{p_{y|x}} && (1.29)\\ &= \mathbb{E}\left[\sum_y p_{y|X} \log \frac{1}{p_{y|X}}\right] && (1.30)\end{aligned}$$

Note that sum inside the expectation is simply another expectation, as in the definition of entropy — but since we only want to apply Jensen's inequality on the outer expectation we spell this one out explicitly. Moreover, by definition of the conditional pmf we have $\mathbb{E}[p_{y|X}] = \sum_x p_x p_{y|x} = \sum_x p_{xy} = p_y$. Hence, using concavity of the entropy as a function of the pmf and Jensen's inequality for the outer expectation, we find

$$\begin{aligned} H(Y|X) &= \mathbb{E}\left[\sum_y p_{y|X} \log \frac{1}{p_{y|X}}\right] && (1.31)\\ &\leq \sum_y \left(\mathbb{E}[p_{y|X}]\right) \log \frac{1}{\mathbb{E}[p_{y|X}]} && (1.32)\\ &= \sum_y p_y \log \frac{1}{p_y} && (1.33)\\ &= H(Y)\,. && (1.34)\end{aligned}$$

Equality in Jensen's inequality only holds if either X is deterministic or if $p_{y|x} = p_y$ for all x and y, but this only holds if X and Y are in fact independent. □

The second relation in Eq. (1.27) can be strengthened by considering three random variables X, Y and Z. In that case, we have

$$H(X|YZ) \leq H(X|Z)\,. \tag{1.35}$$

This is sometimes referred to as *strong sub-additivity*. The proof follows from (regular) sub-additivity, applied to the entropies $H(X|Y, Z = z)$ and $H(X|Z = z)$, and averaging the resulting inequalities.

Question 1.13. *Can you construct a formal proof out of the above sketch?*

1.2.3 Mutual information

We have already established that $H(XY) \neq H(X) + H(Y)$ in general, and hence also $H(Y|X) \neq H(Y)$ by the chain rule. The difference between these two quantities clearly tells us something about how much the uncertainty about Y changes when we learn X, or in other words, about how much information X contains about Y. This leads us to the definition of mutual information,

Definition 1.14. *The* mutual information *between X and Y is defined as*

$$I(X:Y) := H(Y) - H(Y|X) \tag{1.36}$$

It is not evident immediately from the way we defined it here but this expression is symmetric between X and Y. Namely, using the chain rule for conditional entropy (recall Proposition 1.10) twice, we can write

$$I(X:Y) = H(Y) - H(Y|X) = H(Y) + H(X) - H(XY) = H(X) - H(X|Y)\,. \tag{1.37}$$

The mutual information is thus a symmetric measure of the correlation between the two random variables.

Using these various equivalent expressions it is then easy to derive some bounds on the mutual information. First, sub-additivity of the entropy directly implies that $I(X:Y) \geq 0$, so the mutual information is non-negative, and it vanishes only if the two random variables are independent (a consequence of Proposition 1.12). This is consistent with our intuitive notion of information — we cannot know less than nothing after all! We also cannot know more than everything, i.e. the mutual information can never exceed the minimal entropy of its constituent parts.

Question 1.15. *Using the bounds on entropies established in the previous sections, show that $I(X:Y) \leq \log \min\{|\mathcal{X}|, |\mathcal{Y}|\}$. Give an example that saturates the bound.*

Example 1.16. *Consider two binary random variables X and Y with joint pmf*

$$P_{XY}(0,0) = P_{XY}(1,1) = \frac{1}{4}(1+r), \quad P_{XY}(0,1) = P_{XY}(1,0) = \frac{1}{4}(1-r) \tag{1.38}$$

for $r \in [-1,1]$. We can compute the mutual information between X and Y as follows:

$$I(X:Y) = H(X) - H(X|Y) = 1 - h\left(\frac{1+r}{2}\right) \tag{1.39}$$

So this function takes its maximum at $r = -1$ and $r = 1$ and drops to zero for $r = 0$.

Question 1.17. *You might have heard of the correlation coefficient in statistics:*

$$\rho = \frac{\mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])}{\sqrt{\mathrm{Var}(X)\mathrm{Var}(Y)}} \tag{1.40}$$

Can you determine ρ as a function of r?

If we have three random variables X, Y and Z we can ask for the mutual information between X and Y conditioned on knowing Z, the *conditional mutual information*. It is defined as

$$I(X:Y|Z) := \sum_z P_Z(z) I(X:Y|Z=z)\,. \tag{1.41}$$

Various equivalent expressions can then be readily derived, e.g.,

$$I(X:Y|Z) = H(Y|Z) - H(Y|XZ) = H(X|Z) - H(X|YZ)\,. \tag{1.42}$$

Moreover, the *chain rule* for the mutual information states that

$$I(X:YZ) = I(X:Y) + I(X:Z|Y)\,, \tag{1.43}$$

which can be verified by a close inspection of the definition of both conditional and unconditional mutual information.

Consider now the special case where $X - Z - Y$ form a Markov chain. In this case $P_{X|YZ} = P_{X|Z}$ and thus $H(X|YZ) = H(X|Z)$. As a consequence, the conditional mutual information $I(X:Y|Z)$ as written in (1.42) vanishes.

One of the most intriguing properties of the mutual information is the *data-processing inequality* for mutual information. It states that the mutual information can never increase when we apply an operation that only acts on one of the parts. Intuitively this tells us that by manipulating one of the random variables without looking at the other we cannot increase the correlations between the pair.

We can formalise this using the notion of Markov chains.

Proposition 1.18. *Let $X - Y - Z$ form a Markov chain. Then, $I(X:Y) \geq I(X:Z)$.*

Proof. Since $I(X:Z|Y) = 0$, the chain rule for mutual information implies that $I(X:Y) = I(X:YZ)$. It thus remains to show that

$$I(X:Z) \leq I(X:YZ)\,. \tag{1.44}$$

But, since $I(X:Z) = H(X) - H(X|Z)$ and $I(X:YZ) = H(X) - H(X|YZ)$, the relation in Eq. (1.44) is equivalent to the condtion $H(X|Z) \geq H(X|YZ)$, which is in turn ensured by the strong sub-additivity of entropy. □

Question 1.19. *Can you also show that $I(Y:Z) \geq I(X:Z)$ under the same assumption?*

1.3 Relative entropy

The relative entropy appears when we want to compare two different probability distributions. We define it here only for discrete random variables (or rather the respective pmfs), but this can be readily generalised to other probability measures.

Definition 1.20. *Let P and Q be two pmfs on an alphabet $\mathcal{X}$. The* relative entropy *of P with regards to Q is defined as*

$$D(P\|Q) := \sum_{\substack{x\in\mathcal{X}\\ P(x)>0}} P(x)\log\frac{P(x)}{Q(x)}\,. \tag{1.45}$$

if $P(x) > 0 \implies Q(x) > 0$ for all $x \in \mathcal{X}$, and $D(P\|Q) = +\infty$ otherwise.

In the following, instead of restricting the sum, we will use the convention that $0\log\frac{0}{0} = 0$.

We can alternatively see the relative entropy as the expectation value of the *log-likelihood ratio*, namely we can write

$$D(P\|Q) = \mathbb{E}\left[Z(X)\right], \qquad \text{where} \qquad Z(X) = \log\frac{P(X)}{Q(X)} \tag{1.46}$$

and X is distributed according to P. The random variable $Z(X)$ is called the log-likelihood ratio. It takes on the role of the surprisal in the definition of entropy. We will explore this random variable and its distribution much more when we discuss the information spectrum method and hypothesis testing later on.

Just by manipulating the definition, we are able to show the following equivalences.

Proposition 1.21. *Let X and Y be random variables on alphabets $\mathcal{X}$ and $\mathcal{Y}$. Moreover, let U be a uniform random variable on $\mathcal{X}$. Then the following relations are true:*

$$H(X) = \log|\mathcal{X}| - D(P_X\|U_X) \tag{1.47}$$

$$H(X|Y) = \log|\mathcal{X}| - D(P_{XY}\|U_X \times P_Y) \tag{1.48}$$

$$I(X:Y) = D(P_{XY}\|P_X \times P_Y)\,. \tag{1.49}$$

You will prove these equivalences in the homework. They turn out to be very useful because they essentially tell us that once we established properties of the relative entropy this has immediate consequences also for the derived quantities

We will need two important properties of the relative entropy. The first proposition establishes that the relative entropy is always positive.

Proposition 1.22. *For any two pmfs P and Q, we have $D(P\|Q) \geq 0$ with equality if and only if $P = Q$.*

Proof. We can assume without loss of generality that the quantity is finite, as otherwise the statement is trivially true. We first note that $x \mapsto -\log x$ is strictly convex. Hence,

$$D(P\|Q) = \sum_{x:P(x)>0} P(x) \log \frac{P(x)}{Q(x)} \tag{1.50}$$

$$= \sum_{x:P(x)>0} P(x) \left(-\log \frac{Q(x)}{P(x)} \right) \tag{1.51}$$

$$\geq -\log \left(\sum_{x:P(x)>0} P(x) \frac{Q(x)}{P(x)} \right) \tag{1.52}$$

$$= -\log \left(\sum_{x:P(x)>0} Q(x) \right) \tag{1.53}$$

$$\geq -\log \left(\sum_{x} Q(x) \right) = 0\,. \tag{1.54}$$

Equality in the second inequality only holds if P and Q have the same support. Moreover, equality in the first inequality holds if $\frac{Q(x)}{P(x)}$ is independent of x for any x in the support of P. These two statements are both true only if $P(x) = Q(x)$ for all $x \in \mathcal{X}$, and thus $P = Q$. $\square$

An immediate corollary of Propositions 1.21 and 1.22 is that $I(X : Y)$ is positive and zero only if X and Y are independent.

Question 1.23. *Can you see why?*

Finally, there is one property of the relative entropy that implies all other properties of both entropy and mutual information. It states that applying a noisy operation, i.e. a stochastic map or channel, on both arguments of the relative entropy will never increase the relative entropy. Together with the positivity of relative entropy this justifies that we think of it as a measure of similarity or distinguishability. If the relative entropy is small the two pmfs are similar and hard to distinguish by observing the outcomes of a random experiment. Observing the outcomes after further noise has been applied should make distinguishing them even harder, and that is exactly what the *data-processing inequality* for relative entropy tells us.

Proposition 1.24. *Let P_X and Q_X be two pmfs on an alphabet $\mathcal{X}$ (the input distributions), and let $P_{Y|X}$ be a conditional pmf (the channel). Define the marginals (the output distributiions)*

$$P_Y(y) = \sum_{x\in\mathcal{X}} P_{Y|X}(y|x) P_X(x) \qquad \textit{and} \qquad Q_Y(y) = \sum_{x\in\mathcal{X}} P_{Y|X}(y|x) Q_X(x)\,. \tag{1.55}$$

Then, the data-processing inequality *(DPI) states that*

$$D(P_X\|Q_X) \geq D(P_Y\|Q_Y)\,. \tag{1.56}$$

Proof. Consider now the joint distributions $P_{XY}(x,y) = P_{Y|X}(y|x)P_X(x)$ and $Q_{XY}(x,y) = P_{Y|X}(y|x)Q_X(x)$, using the usual shorthand notation for conditional and marginal distributions. We first show that

$$D(P_{XY}\|Q_{XY}) - D(P_Y\|Q_Y) = \left(\sum_{x,y} p_{xy}\log\frac{p_{xy}}{q_{xy}}\right) - \left(\sum_{y} p_{y}\log\frac{p_{y}}{q_{y}}\right) \tag{1.57}$$

$$= \sum_{x,y} p_{xy}\left(\log\frac{p_{xy}}{q_{xy}} - \log\frac{p_y}{q_y}\right) \tag{1.58}$$

$$= \sum_{y} p_y \sum_x p_{x|y}\log\frac{p_{x|y}}{q_{x|y}} \tag{1.59}$$

$$= \sum_y p_y\, D(P_{X|Y=y}\|Q_{X|Y=y}) \geq 0\,, \tag{1.60}$$

where we have used the positivity of relative entropy in the last step. Similarly, we have

$$D(P_{XY}\|Q_{XY}) - D(P_X\|Q_X) = \sum_x p_x\, D(P_{Y|X=x}\|Q_{Y|X=x}) = 0 \tag{1.61}$$

since $Q_{Y|X} = P_{Y|X}$ by construction of the joint distribution. Combining Eqs. (1.57)–(1.60) and (1.61) yields the desired inequality. □

It turns out that all the properties of entropy, conditional entropy and mutual information we discussed previously can be derived form the DPI. As an example we give here a strengthening of the strong sub-additivity, which we call the data-processing inequality for conditional entropy. It intuitively states that any processing of the side information can at most increase the conditional entropy.

Corollary 1.25. *Let P_{XY} be a joint pmf and $P_{Z|Y}$ a conditinal pmf. Define now the pmf*

$$P_{XZ}(x,z) = \sum_y P_{XY}(x,y)P_{Z|Y}(z|y)\,. \tag{1.62}$$

Then, we have $H(X|Y) \leq H(X|Z)$.

Proof. Let us express the inequality in terms of relative entropies using Proposition 1.21. This reads

$$\log|\mathcal{X}| - D(P_{XY}\|\, U_X \times P_Y) \leq \log|\mathcal{X}| - D(P_{XZ}\|U_X \times P_Z)\,. \tag{1.63}$$

or simply $D(P_{XY}\|\, U_X \times P_Y) \geq D(P_{XZ}\|U_X \times P_Z)$. But this is imply the DPI applied to the channel $P_{Z|Y}$ that happens to leave X untouched. □

Chapter 2

Source coding

[Week 3–5]

Intended learning outcomes:

- You can determine if a code is instantaneous and if the codeword lengths are optimal using the McMillan–Kraft inequality.
- You can evaluate the quality of a variable-length code by comparing it to the fundamental limits.
- You can construct a Huffman code for any discrete source, and understand the algorithm and its properties.
- You understand the mathematical model used to study block codes asymptotically, and can compute the code rate.
- You understand Fano's inequality and typical sets and how they can be used to derive the fundamental limits of compression.

Book reference: Chapter 5 in Cover & Thomas [**?**].

2.1 Problem setup and definitions

In this chapter we are concerned with removing redundancy. In the first section we will introduce the formal mathematical model we use to investigate source coding, or compression.

2.1.1 Data source

We are given data as a sequence of symbols, for example these could be numbers, letters, colours of pixels, etc., and we would like to store that data in a (preferably short) sequence of bits. (We could generalise to larger alphabets, but conceptually nothing changes so we restrict ourselves to bits here to simplify presentation.) We start by formally defining what we mean by a *data source*, or simply *source* in the remainder of this chapter.

Definition 2.1. *A* data source *is an infinite sequence of random variables*

$$\boldsymbol{X} = X_1, X_2, \ldots, X_k, \ldots . \tag{2.1}$$

- *A source is called* discrete *if the random variables are discrete, i.e. if the source outputs in each iteration $i \in \mathbb{N}$ values from a finite set $\mathcal{X}$.*
- *A source is furthermore called* memoryless *if the X_i are independent and identically distributed (i.i.d.) according to the same pmf P_X, i.e., if we have*

$$P_{X_1 X_2 \ldots X_k \ldots}(x_1, x_2, \ldots, x_k, \ldots) = P_X(x_1) P_X(x_2) \ldots P_X(x_k) \ldots . \tag{2.2}$$

Memoryless here refers to the fact that the distribution of X_i does not depend on the value of X_{i-1} or any other symbol in the sequence, or, formally, $P_{X_i|X_1X_2\ldots X_{i-1}} = P_{X_i} = P_X$. We will not consider sources that output continuous values in this module.

Question 2.2. *Can you see why we cannot expect to store and perfectly recover a continuous variable using finite (digital) memory?*

For most of our theoretical analysis, we will consider a *discrete memoryless source (DMS)*. An example of such a source is the sequence of face values one gets by throwing the same (fair or unfair) die repeatedly. Generally, the assumption that a source is memoryless is simplifying the analysis but in fact most sources do not satisfy this exactly. For example, think of a book (written in English) as a sequence of letters and a source reproducing them one by one. If $X_{i-1} =$ 'q', then $X_i =$ 'u' with much higher probability than the frequency of 'u' in English text would otherwise suggest. Hence, this source is far from memoryless and the corresponding distribution of the random variable is not i.i.d.. Nonetheless, understanding the simple case of discrete memoryless sources properly will allow us to get an intuition for more loosely structured sources as well. Various more complicated models of sources have been analysed in the literature.

2.1.2 Source codes

Next we introduce *source codes*, or simply *codes* for the remainder of this chapter.

Definition 2.3. *A* source code *is a map e and that maps outputs of the source $x \in \mathcal{X}$ to bit strings of variable length, $C(x) \in \{0,1\}^*$. We denote by $\ell(x)$ the length of the* codeword *$C(x)$.*

- *A code is called a* fixed-length *code if $\ell(x) = \ell$ is constant, otherwise it is called a* variable-length *code.*

(NS) A code is called non-singular *if C is injective, i.e. if every $x \in \mathcal{X}$ is mapped to a unique bit string.*

(P) *A code is called a* prefix code *if for any pair $x, x' \in \mathcal{X}$ with $x \neq x'$, the codeword $C(x)$ is not a prefix of the codeword $C(x')$.*[1]

(U) *A code is* uniquely decodable *if there exists a decoder that, for any $n \in \mathbb{N}$ and any sequence $x^n \in \mathcal{X}^n$, can uniquely recover x^n from the bit string $C(x_1)C(x_2)\dots C(x_n)$.*

(I) *A code is* instantaneous *if it is uniquely decodable and if the decoder can deduce the k-th symbol x_k as soon it has seen the bit string $C(x_1)C(x_2)\dots C(x_k)C(x_{k+1})\dots$ up to and including all of $C(x_k)$, even if there is no guarantee that the string is complete.*

Let us note that the codes we consider here are not as general as they could be. In fact, we could also consider codes that take a variable length sequence of input symbols to a codeword (of either fixed or variable length). Such codes are in fact often used in practical applications. A prominent example is the Lempel–Ziv–Markov algorithm for lossless compression, which uses a dictionary to replace often reoccurring variable-length sequences with shorter codewords.

Let us now discuss some of the interrelations between all these code properties. Clearly, any uniquely decodable code is non-singular by definition. However, we observe that not every non-singular code is uniquely decodable. Consider a code on $\mathcal{X} = \{0, 1, 2, 3\}$ that yields the binary representation

$$C(0) = 0, \quad C(1) = 1, \quad C(2) = 10, \quad C(3) = 11\,. \tag{2.3}$$

This code is non-singular but not a prefix code. The codeword string 110 could either be produced by the source sequence $(3, 0)$, by $(1, 2)$ or even by $(1, 1, 0)$, so there is no way for a decoder to distinguish between these three cases.

Proposition 2.4. *A code is instantaneous if and only if it is a prefix code*

Proof. We first show that a prefix code is instantaneous by constructing a decoder. The decoder will read the sequence $C(x_1)C(x_2)\dots$ bit by bit. Once $C(x_1)$ is fully read we can immediately deduce that the first source symbol was x_1 since $C(x_1)$ cannot be a prefix for a longer codeword. Similarly, with this rule in mind, since there is no other codeword that is a prefix to $C(x_1)$ we can be assured that this is indeed the first symbol we will decode. The same procedure continues for the remainder of the string with $C(x_2)C(x_3)\dots$. We know where every codeword ends and can decode them individually and instantaneously.

To verify the other direction, simply note that if a decoder can decide instantly once it has seen the codeword $C(x)$ this implies that $C(x)$ cannot be a prefix to any other codeword $C(x')$. Since this is true for any $x \neq x'$, the code must be a prefix code. □

Thus, any prefix code is uniquely decodable. However, the converse is not true in general, i.e. not every uniquely decodable code is a prefix code. Consider as an example the code

$$C(\text{'a'}) = 1, \quad C(\text{'b'}) = 10, \quad C(\text{'c'}) = 00\,. \tag{2.4}$$

[1] We say that a bit string is a prefix of another bit string if the latter starts with the former, e.g. '01' is a prefix to '0100'.

After seeing 10 we cannot decide if the first symbol was ‘a’ or ‘b’ even though we have seen the full codeword of the first symbol, hence this code is not instantaneous. However, once we have seen a full sequence of codewords, for example 100, we can decode uniquely by looking at the parity of the number of 0’s between two 1’s.

Question 2.5. *Can you come up with a formal decoder for this code?*

Codes can be conveniently represented by binary trees. Binary trees are connected graphs without cycles (trees) where each node (except the root) has exactly one parent and either zero (in which case it is called a leaf) or two children. The two branches emanating from the root and each node are assigned values ‘0’ or ‘1’ and codewords are composed by following a path from the root to a node.

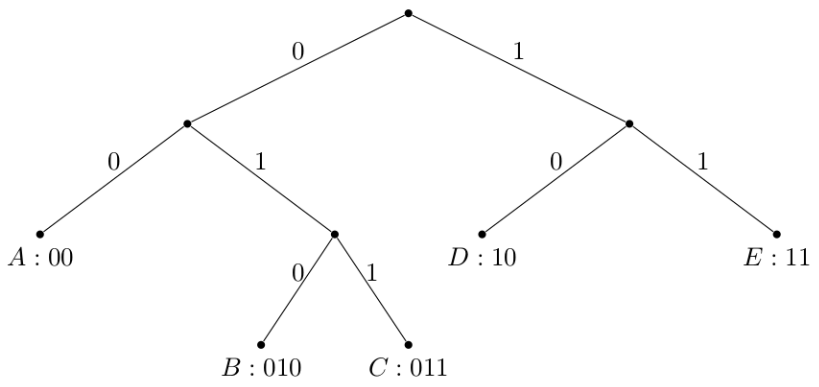


Figure 2.1: Example of a code tree.

The codeword length is equivalent to the depth (i.e. the distance from the root) of the node in the three. For a fixed-length code all the codewords are at the same depth (or level) of the tree. A code is a prefix code if and only if all the codewords are on leaves of the tree.

Question 2.6. *Can you see why this is the case?*

The next two sections will be devoted to variable-length and fixed-length block codes (defined later), respectively.

2.2 Variable-length codes

Before we discuss particular codes, we first want to establish some fundamental limits all codes need to satisfy.

2.2.1 Optimal codeword lengths

The Kraft inequality gives a lower bound on the lengths of codewords in any instantaneous code. It is the first fundamental limit we will establish, it shows us that no code with shorter codeword lengths can exist, and thus if a code achieves equality in (2.5) we know it is optimal in this regard. We present a slightly more general result, the MacMillan–Kraft inequality, which applies for any uniquely decodable code (and not just prefix codes).

Proposition 2.7 (McMillan–Kraft inequality)**.** *Any uniquely decodable code must satisfy the inequality*

$$\sum_{x\in\mathcal{X}} 2^{-\ell(x)} \leq 1\,. \tag{2.5}$$

Conversely, given a set of codewords lengths satisfying Eq. (2.5)*, it is possible to construct a prefix code with these lengths.*

We show the inequality only for instantaneous codes, and the general proof for uniquely decodable (not necessarily instantaneous) codes will be covered in the homework.

Proof. We use the one-to-one correspondence of prefix codes with binary trees for which the codewords are leaves. For any tree we may assign to every node a value 2^{-d} where d is the depth in the tree. The root thus gets the value 1. To show the Kraft inequality we simply need to show that the sum of all the leaf values in a tree cannot exceed 1. To see that this is correct, simply note that by our construction any parent node's value is simply the sum of its children's values, so as we build up the binary tree from the root the sum of the values on all leaves is always exactly 1.

Conversely, given a set of codeword lengths satisfying the inequality, we can always create a binary tree with leaves at the corresponding depths and populate the leaves with codewords. If the inequality is strict there will be unused leaves in the tree. □

2.2.2 Optimal expected codeword length

Finding codewords with short lengths is however only half of the problem — we also need to assign those codewords to elements of $\mathcal{X}$. And we want to do this in such a way as to minimise the expected length of the codeword. That is, for a code $C(x)$ with codeword lengths $\ell(x)$, we define the *expected codeword length* as

$$\bar{\ell}(C) := \sum_{x} P_X(x)\ell(x) = \mathbb{E}[\ell(X)] \tag{2.6}$$

Using the McMillan–Kraft inequality from Proposition 2.7, we can lower bound $\bar{\ell}(C)$ with the entropy $H(X)$ for any uniquely decodable code.

Proposition 2.8. *For any uniquely decodable code C for a discrete source X with distribution P_X, we have*

$$\bar{\ell}(C) \geq H(X)\,. \tag{2.7}$$

Moreover, the equality is saturated if only if the codeword lengths saturate the McMillan-Kraft inequality and $P_X(x) = 2^{-\ell_x}$ for some set of numbers $\ell_x \in \mathbb{N}$.

Proof. We evaluate

$$\begin{aligned}
\bar{\ell}(C) - H(X) &= \mathbb{E}\left[\ell(X) - \log \frac{1}{P_X(X)}\right] && (2.8)\\
&= \mathbb{E}\left[\log \frac{P_X(X)}{2^{-\ell(X)}}\right] && (2.9)\\
&\geq \mathbb{E}\left[\log \frac{t \cdot P_X(X)}{2^{-\ell(X)}}\right] && (2.10)\\
&= D(P_X \| Q_X) && (2.11)\\
&\geq 0, && (2.12)
\end{aligned}$$

where we introduced the constant $t = \sum_x 2^{-\ell(x)} \leq 1$ (by the McMillan–Kraft inequality) and the pmf $Q(x) = \frac{1}{t} \cdot 2^{-\ell(x)}$. The final inequality is simply due to the non-negativity of relative entropy.

If the conditions for saturation are met we can see that the two inequalities become equalities as $t = 1$ and $P_X = Q_X$ if we choose $\ell(x) = \ell_x$. Conversely, using the positive definiteness of the relative entropy, we see that these conditions are in fact necessary to achieve equality. □

2.2.3 Shannon code

The above result allows us to show that certain codes have optimal expected codeword lengths. For example for the source X that outputs symbols 'a', 'b' and 'c' with probabilities $\frac{1}{2}$, $\frac{1}{4}$ and $\frac{1}{4}$, respectively, the code

$$C(\text{'a'}) = 0, \quad C(\text{'b'}) = 10, \quad C(\text{'c'}) = 11 \tag{2.13}$$

satisfies $\bar{\ell}(C) = \frac{1}{2} + 2 \cdot \frac{1}{4} \cdot 2 = \frac{3}{2}$ and $H(X) = \frac{1}{2}\log 2 + 2 \cdot \frac{1}{4} \log 4 = \frac{3}{2}$, and thus, we know that it is optimal thanks to Proposition 2.8.

The above example is constructed in such a way that all the probabilities are negative powers of 2 and the codeword lengths satisfy the Kraft inequality with equality, and in this particular case it is easy to see from the proof of Proposition 2.8 that $\bar{\ell}(C) = H(X)$. If the probabilities do not have this form the same construction does not generally work. However, we can show the following.

Proposition 2.9. *For a discrete source X with distribution P_X there always exists a prefix code C with $\bar{\ell}(C) \leq H(X) + 1$.*

The code we construct to proof this is called the Shannon code, and it not optimal in general. However, the this bound, together with Proposition 2.8, shows that we lose at most 1 bit per symbol using this code.

Proof. We can choose codeword lengths $\ell(x) = \left\lceil \log \frac{1}{P_X(x)} \right\rceil$. These satisfy the Kraft inequality since

$$\sum_x 2^{-\ell(x)} = \sum_x 2^{-\left\lceil \log \frac{1}{P_X(x)} \right\rceil} \leq \sum_x 2^{-\log \frac{1}{P_X(x)}} = \sum_x P_X(x) = 1\,. \tag{2.14}$$

Hence, using Proposition 2.7 we may construct a prefix codes using these lengths. Moreover, for this code we have

$$\begin{aligned} \bar{\ell}(C) &= \sum_x P_X(x) \left\lceil \log \frac{1}{P_X(x)} \right\rceil && (2.15)\\ &\leq \sum_x P_X(x) \left(\log \frac{1}{P_X(x)} + 1 \right) && (2.16)\\ &= H(X) + 1\,. && (2.17) \end{aligned}$$

□

2.2.4 Huffman codes

In this section we will construct prefix codes with optimal expected codeword length, so-called Huffman codes. We will first learn how to construct the codes and then use this construction to show optimality. Interestingly, the codes were invented by a student that was in the same situation as you are right now!

> In 1951, David Huffman and his MIT information theory classmates were given the choice of a term paper or a final exam. The professor, Robert Fano, assigned a term paper on the problem of finding the most efficient binary code. Huffman, unable to prove any codes were the most efficient, was about to give up and start studying for the final when he hit upon the idea of using a frequency-sorted binary tree and quickly proved this method the most efficient. In doing so, Huffman outdid Fano, who had worked with information theory inventor Claude Shannon to develop a similar code.

The code is constructed using Algorithm 2.1, which step-by-step merges a forest of trivial binary trees into a single binary tree.

The construction is not unique because in each step we can assign the labels '0' and '1' in either way to the two children. Moreover, we are asked to select the two trees with smallest probabilities, but there might be different such pairs, e.g. if we start with a source X with

```
Input: List of symbols $x \in \mathcal{X}$ with probabilities $p_x = P_X(x)$
Output: Binary tree for the Huffman code

% initialise forest
for each $x \in \mathcal{X}$ do
    Create a tree with a root node labelled by the probability $p_x$ (and the symbol $x$) and no
    other nodes;
    Add this tree to the forest;
end
% condense forest into a single tree
while number of trees in the forest are larger than 1 do
    select the two trees whose roots have the smallest probabilities, say $p$ and $p'$;
    join the two trees by adding a new root with probability $p + p'$ with the two trees as
    children, the edges are labelled with '0' and '1';
end
return last remaining tree in the forest;
```

Algorithm 2.1: Construction of a Huffman code tree.

symbols and probabilities (‘a’, $\frac{1}{3}$), (‘b’, $\frac{1}{3}$), (‘c’, $\frac{1}{6}$), (‘d’, $\frac{1}{6}$) then both of these codes, C_1 and C_2, are possible Huffman codes:

$$C_1(\text{‘a’}) = 00, \qquad C_1(\text{‘b’}) = 01, \qquad C_1(\text{‘c’}) = 10, \qquad C_1(\text{‘d’}) = 11 \tag{2.18}$$

$$C_2(\text{‘a’}) = 0, \qquad C_2(\text{‘b’}) = 10, \qquad C_2(\text{‘c’}) = 110, \qquad C_2(\text{‘d’}) = 111 \tag{2.19}$$

$$\tag{2.20}$$

Question 2.10. *Can you retrace how they are created step-by-step?*

The codeword lengths for both codes are optimal according to the Kraft inequality, that is, we have

$$\sum_x 2^{-\ell(x)} = 4 \cdot 2^{-2} = 1 \qquad \text{and} \tag{2.21}$$

$$\sum_x 2^{-\ell(x)} = 2^{-1} + 2^{-2} + 2 \cdot 2^{-3} = 1 \tag{2.22}$$

for C_1 and C_2, respectively. We can further compute their respective expected codeword lengths. This yields

$$\bar{\ell}(C_1) = 2 \tag{2.23}$$

$$\bar{\ell}(C_2) = \frac{1}{3} \cdot 1 + \frac{1}{3} \cdot 2 + \frac{1}{6} \cdot 3 + \frac{1}{6} \cdot 3 = 2 \tag{2.24}$$

Let us compare this to the entropy $H(X) = 2 \cdot \frac{1}{3} \log 3 + 2 \cdot \frac{1}{6} \log 6 \approx 1.92$. So from the entropy bound alone we cannot deduce that these codes are optimal in terms of the expected codeword length — but they in fact are!

Proposition 2.11. *Given a source X with probability distribution P_X, any code constructed using Algorithm 2.1 achieves the minimal expected codeword length for any prefix code.*

We call such a code with minimal expected length an *optimal prefix code.* The proof relies on the following lemma, which we show first.

Lemma 2.12. *There exists an optimal prefix code with the following property:*

- *The two longest codewords are siblings and their respective source symbols have the two smallest probabilities (this is not always unique).*

Proof. An optimal code always corresponds to a binary tree with no unused leaves — if not we can compress the tree by removing the parent of the unused leaf, reducing the expected codeword length. There is always at least one pair of leaves at maximum depth, and those are thus occupied with codewords. If those codewords would not correspond to the two symbols with smallest probability we could exchange symbols to move them there, a process which clearly cannot increase the expected codeword length. □

```
RecursiveHuffman:
Input: Forest $f_{\text{in}}$
Output: Forest $f_{\text{out}}$

if number of trees in $f_{\text{in}}$ = 1 then
    return $f_{\text{out}} = f_{\text{in}}$;
else
    select the two trees in $f_{\text{in}}$ whose roots have the smallest probabilities, say $p$ and $p'$;
    create new forest $f'$ from $f_{\text{in}}$ by joining the selected trees as in Algorithm 2.1;
    return $f_{\text{out}}$ = RecursiveHuffman($f'$);
end
```

Algorithm 2.2: Recursive formulation of the Huffman algorithm.

Proof of Proposition 2.11. The recursive formulation of the Huffman algorithm in Algorithm 2.2 is useful here. Clearly the algorithm produces an optimal code when $|\mathcal{X}| = 2$ since in this case the optimal code simply assigns the codewords 0 and 1 to the two symbols, and this is exactly what the output of the Huffman algorithm will be.

We will thus prove optimality by induction as follows. Let us, without loss of generality, label elements such that our source symbols have probabilities $p_1 \geq p_2 \geq \ldots \geq p_n$ and ordered such that the Huffman algorithm will pick p_{n-1} and p_n as the smallest elements if there are ambiguities. By induction we may assume that the recursive Huffman algorithm provides us with an optimal tree when we call it for $n-1$ symbols with the probabilities $p_1, \ldots, p_2, \ldots, p_{n-2}, p_{n-1} + p_n$. We denote the expected codeword length of this optimal tree by $L^*_{n-1}(p_1, \ldots, p_2, \ldots, p_{n-2}, p_{n-1} + p_n)$.

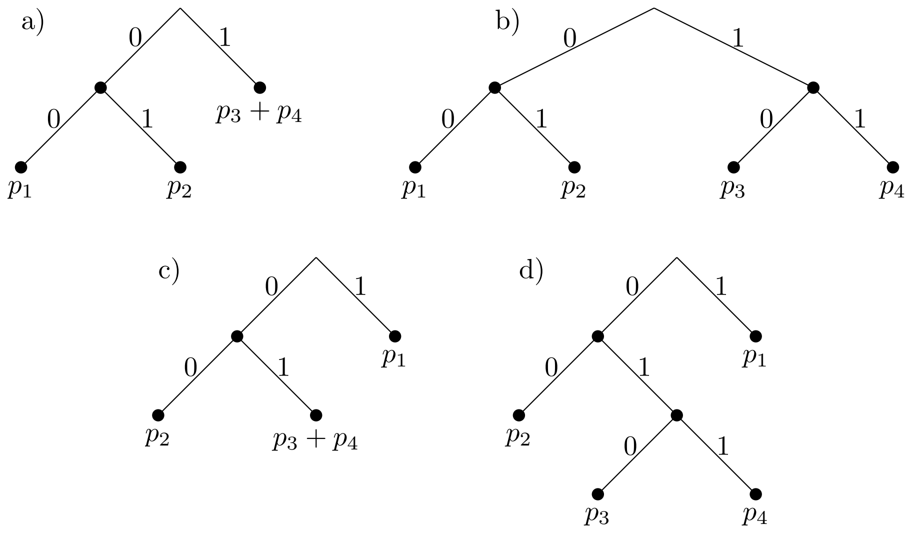


Figure 2.2: **Optimality of Huffman coding with 4 symbols, recursive step.** Assume $p_1 \geq p_2 \geq p_3 \geq p_4$. We are given a minimal length code for $(p_1, p_2, p_3 + p_4)$, which must be either of the form a) if $p_3 + p_4 \geq p_1$ or of the form c) if $p_3 + p_4 \leq p_1$. If equality holds both solutions are optimal. The trees produced by the Huffman algorithm for (p_1, p_2, p_3, p_4) (assuming by induction it produces an optimal tree for $(p_1, p_2, p_3 + p_4)$), are thus as in b) and d), respectively. In both cases, we have $L_4 = L_3^* + p_3 + p_4$.

The Huffman algorithm for n symbols, by definition in its recursive form, produces exactly this tree but with the the $(n-1)$-th node split into two siblings with probabilities p_{n-1} and p_n (see Figure 2.2 for an example). Its expected codeword length, L_n, thus satisfies

$$L_n = L_{n-1}^*(p_1, \ldots, p_2, \ldots, p_{n-2}, p_{n-1} + p_n) + p_{n-1} + p_n \,. \tag{2.25}$$

Note that we added $p_{n-1} + p_n$ as compared to the tree for $n-1$ symbols those two leaves are now one level deeper, which increases the codeword length by 1 with probability $p_{n-1} + p_n$.

On the other hand, we have

$$L_{n-1}^*(p_1, \ldots, p_2, \ldots, p_{n-2}, p_{n-1} + p_n) \leq L_n^*(p_1, \ldots, p_2, \ldots, p_{n-1}, p_n) - p_{n-1} - p_n \tag{2.26}$$

since a valid (although not necessarily optimal) prefix code for the $n-1$ probabilities can be constructed from an optimal prefix code for n symbols by merging the two leaves at maximum depth with minimal probability (which exist due to Lemma 2.12) into a single leaf. Such a code has expected codeword length $L_n^* - p_{n-1} - p_n$, and thus in particular the optimal length of such a tree for $n-1$ symbols must satisfy $L_{n-1}^* \leq L_n^* - p_{n-1} - p_n$.

Combining Eqs. (2.25) and (2.26) yields $L_n \leq L_n^*(p_1, \ldots, p_2, \ldots, p_{n-1}, p_n)$, and since L_n can never be smaller than the optimal codeword length (by definition of the latter), these two quantities must in fact be equal. This proves that the Huffman code construction is optimal. □

2.3 Fixed-length block codes

Fixed-length codes have the property that all codewords are equally long. If we require error-free compression, then there is not much flexibility: the expected codeword length has to be equal to $\lceil \log |\mathcal{X}| \rceil$ (assuming that every source symbol appears with strictly positive probability). The picture gets dramatically more interesting if we encode a whole block of n source symbols and only require that the probability of a decoding error vanishes as $n \to \infty$. A *block code* of length n takes a sequence of n source outputs $x^n \in \mathcal{X}^n$ as input and outputs a binary string $C(x^n)$.

2.3.1 Setup for block coding

As we are observing a long sequence of symbols $X_1, X_2, \ldots, X_k, \ldots$, one thing we can do is to treat a block of, say n, symbols as a single symbol (with a much larger alphabet of size $|\mathcal{X}|^n$) and then try to find efficient codes for such blocks. Obviously then we can no longer encode and decode instantaneously as we will need to wait for the full block to perform the encoding, and the decoding operation will in turn yield a full block as well.

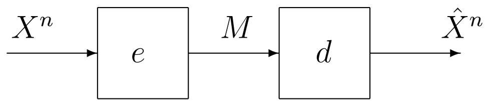


Figure 2.3: Illustration of the fixed-to-fixed length source coding problem.

So for a block code, we observe a sequence of random variables $X^n = (X_1, \ldots, X_n)$ from a discrete memoryless source and we would like to compress it into a random variable $M \in \{0,1\}^L$, the codeword, using an encoder, a function e from X^n to M. Later on, the decoder d will produce an estimate $\hat{X}^n$ of X^n from M. See Fig. 2.3 for an illustration. If we demand that

$$P(\hat{X}^n \neq X^n) = 0 \tag{2.27}$$

then M needs to take on at least

$$\left|\left\{x \in \mathcal{X} : P_X(x) > 0\right\}\right|^n \tag{2.28}$$

different values, and thus we need to choose $L \geq \left\lceil n \log \left|\left\{x \in \mathcal{X} : P_X(x) > 0\right\}\right| \right\rceil$.

Question 2.13. *Argue why this is sufficient and optimal.*

Comparing this to the bound $L \geq n \left\lceil \log \left|\left\{x \in \mathcal{X} : P_X(x) > 0\right\}\right| \right\rceil$, which we would have arrived at by encoding each source symbol separately using a fixed length code, we see that there is some improvement. However, it is at most n bits. Can we do better if we relax the stringent condition in (2.27) to be such that

$$P(\hat{X}^n \neq X^n) \leq \epsilon \tag{2.29}$$

for any $\epsilon > 0$ positive but arbitrarily small? Let us formalise this.

Definition 2.14 (Block code). *An* $(n, 2^L)$-fixed-length source code *(or simply an* $(n, 2^L)$*-code) consists of an* encoder*,* e*, and a* decoder*,* d*, where*

- $e : \mathcal{X}^n \to \{0,1\}^L$ *and*
- $d : \{0,1\}^L \to \mathcal{X}^n$

The number n is called the *block length* of the code; L is the length of the codeword; and $R = \frac{L}{n}$ is called the rate of the code. The rate simply evaluates how many bits of codeword this codes uses per symbol of the source to store its output.

Definition 2.15 (Achievable rate). *A rate* R *is achievable for a DMS* $\boldsymbol{X}$ *if there exists a sequence of* $(n, 2^{\lfloor nR \rfloor})$*-codes for* $n \in \mathbb{N}$ *with encoder* e_n *and decoder* d_n *such that*

$$\lim_{n\to\infty} P(\hat{X}^n \neq X^n) = 0 \tag{2.30}$$

where

$$\hat{X}^n = d_n(M), \quad \textit{and} \quad M = e_n(X^n) \tag{2.31}$$

are the reconstructed source and the codeword, respectively.

Question 2.16. *Show that if* R *is achievable so is any* $R' \geq R$*.*

Thus, what we really are interested in is the smallest R that is still achievable.

Definition 2.17 (Optimal source coding rate). *The optimal source coding rate for the DMS* $\boldsymbol{X}$*, denoted as* $R^*(\boldsymbol{X})$*, is defined to be the infimum of all achievable rates, i.e.,*

$$R^*(\boldsymbol{X}) = \inf\{R : R \text{ is achievable}\}. \tag{2.32}$$

Finding the optimal source coding rate looks like a formidable problem to solve at first sight. Note that the notion of achievable rate is asymptotic, namely we need to consider a sequence of codes, a code for each $n \in \mathbb{N}$, so that it is not even obvious that $R^*(\boldsymbol{X})$ is computable in finite time from a complexity-theoretic perspective. However, in his seminal work Shannon [**?**] showed that $R^*(X)$ has a simple form for a DMS.

Theorem 2.18 (Fixed-length data compression). *For any DMS* $\boldsymbol{X}$ *with pmf* P_X*, we have*

$$R^*(\boldsymbol{X}) = H(X) \tag{2.33}$$

To prove that $R^*(\boldsymbol{X}) = H(X)$, we must prove the *achievability part*, $R^*(\boldsymbol{X}) \leq H(X)$, and the *converse part*, $R^*(\boldsymbol{X}) \geq H(X)$.

- Achievability means that, for every $R > H(X)$, we must exhibit a sequence of $(n, 2^{\lfloor nR \rfloor})$-codes such that the error probability vanishes as $n \to \infty$.
- The converse implies that we cannot do better than this, i.e., there is no sequence of $(n, 2^{\lfloor nR \rfloor})$-codes where $R < H(X)$ such that we have a vanishing error probability.

In most problems in information theory the two proofs (achievability and converse) use quite different techniques and we thus threat them separately.

2.3.2 Proof of converse and Fano's inequality

For the converse we will use Fano's inequality, which is a fundamental tool in the analysis of information processing tasks. We formulate it here as a statement about conditional entropies of strongly correlated random variables.

Lemma 2.19 (Fano's inequality)**.** *Let X, Y be random variables with joint pmf P_{XY} and let $\epsilon := P[X \neq Y]$. Then*

$$H(X|Y)_P \leq h(\epsilon) + \epsilon \log(|X| - 1) \leq 1 + \epsilon \log |X| \,, \tag{2.34}$$

where $h(\epsilon) = -\epsilon \log \epsilon - (1-\epsilon)\log(1-\epsilon)$ is the binary entropy.

This essentially tells us that if the probability that the two random variables differ is small, then so is the conditional entropy. We should expect this, since the conditional entropy measures the uncertainty of X given side information Y: if ϵ is small then knowing the value y of Y allows us to guess that $X = y$ as well, which is correct with high probability.

Proof. First, we recall that $H(X|Y)_P = \sum_y P_Y(y) H(X|Y=y)$. We will bound these terms individually first. Define $\epsilon_y = 1 - P_{X=y|Y=y}$ so that $\sum_y P_Y(y)\epsilon_y = \epsilon$. Then, we find

$$\begin{aligned}
H(X|Y=y) &= -\sum_x P_{X|Y=y}(x) \log P_{X|Y=y}(x) && (2.35)\\
&= -(1-\epsilon_y)\log(1-\epsilon_y) - \sum_{x\neq y} P_{X|Y=y}(x) \log P_{X|Y=y}(x) && (2.36)\\
&= -(1-\epsilon_y)\log(1-\epsilon_y) - \epsilon_y \log \epsilon_y - \epsilon_y \sum_{x \neq y} \frac{P_{X|Y=y}(x)}{\epsilon_y} \log \frac{P_{X|Y=y}(x)}{\epsilon_y} && (2.37)\\
&= h(\epsilon_y) - \epsilon_y \sum_{x\neq y} \frac{P_{X|Y=y}(x)}{\epsilon_y} \log \frac{P_{X|Y=y}(x)}{\epsilon_y} && (2.38)\\
&\leq h(\epsilon_y) + \epsilon_y \log(|X| - 1) && (2.39)
\end{aligned}$$

In the last step we simply used that the entropy is upper bounded by the logarithm of the support as shown in Chapter 1 .

It remains to take an average of the above bound. Using (once again) concavity of the entropy, we find

$$\begin{aligned}
\sum_y P_Y(y) H(X|Y=y) &\leq \sum_y P_Y(y) h(\epsilon_y) + \epsilon_y \log |X| && (2.40)\\
&\leq h\Big(\sum_y P_Y(y)\epsilon_y\Big) + \epsilon \log |X| && (2.41)\\
&= h(\epsilon) + \epsilon \log |X| \,. && (2.42)
\end{aligned}$$

Finally, we can bound $h(\epsilon) \leq \log 2 = 1$ and $|X| - 1 \leq |X|$ to make the bound a bit simpler but still sufficient for most purposes. □

Proof of converse of Theorem 2.18. Consider now any sequence of $(n, 2^{\lfloor nR \rfloor})$-codes with encoders e_n and decoders d_n that satsify $\epsilon_n \to 0$ as $n \to \infty$, where

$$\epsilon_n := P(\hat{X}^n \neq X^n) \tag{2.43}$$

with $\hat{X}^n = d_n(e_n(X^n))$. Fano's inequality applied to the estimation of X^n yields

$$H(X^n|\hat{X}^n) \leq \epsilon_n n \log |\mathcal{X}| + 1 \tag{2.44}$$

Since $H(X^n|M) \leq H(X^n|\hat{X}^n)$ by the data-processing inequality for the conditional entropy (see Corollary 1.25) applied to the decoder d_n, this can be relaxed to

$$H(X^n|M) \leq \epsilon_n n \log |\mathcal{X}| + 1 \tag{2.45}$$

Furthermore, using the dimension bound for $|M| = 2^{\lfloor nR \rfloor} \leq 2^{nR}$, and the definition of mutual information, we find

$$\begin{aligned} nR &\geq H(M) && (2.46)\\ &= I(X^n : M) + H(M|X^n) && (2.47)\\ &= I(X^n : M) && (2.48)\\ &= nH(X) - H(X^n|M) && (2.49) \end{aligned}$$

We can now apply Eq. (2.45) to get

$$R \geq H(X) - \epsilon_n \log |\mathcal{X}| - \frac{1}{n}\,. \tag{2.50}$$

Since this inequality must hold for large n and $\epsilon_n \to 0$ as $n \to \infty$, we thus must have that

$$R \geq H(X)\,. \tag{2.51}$$

Moreover, since this holds for any sequence of codes, we conclude that $R^*(\boldsymbol{X}) \geq H(X)$. $\square$

2.3.3 Proof of achievability and typical sets

The main idea in the proof of achievability is to only encode sequences of source outputs that are "typical" in a sense we will make precise below. The sets of typical sequences are chosen in such a way that two crucial properties hold:

- There are not too many such sequences so that we can encode them efficiently.
- We can safely ignore all the sequences that are not typical since they are guaranteed to only occur with very low probability.

Let us now make this more formal.

Definition 2.20. *Let $\epsilon \in (0,1)$ and consider a DMS $\boldsymbol{X}$. The ϵ-typical set for $\boldsymbol{X}$, for each $n \in \mathbb{N}$, is defined as*

$$A_\epsilon^{(n)}(\boldsymbol{X}) := \left\{ x^n \in \mathcal{X}^n : \left| \frac{1}{n} \log \frac{1}{P_{X^n}(x^n)} - H(X) \right| \leq \epsilon \right\} \tag{2.52}$$

where

$$P_{X^n}(x^n) = P(X^n = x^n) = \prod_{i=1}^{n} P_X(x_i), \qquad \forall\, x^n \in \mathcal{X}^n. \tag{2.53}$$

We call the elements of $A_\epsilon^{(n)}(\boldsymbol{X})$ *ϵ-typical sequences* of length n for the source $\boldsymbol{X}$. In words, this means that the typical sequences are those whose negative log-likelihood (per symbol) is very close to the entropy of X, the i.i.d. output of the source. Note that

$$H(X) = \frac{1}{n} H(X_1 X_2 \ldots X_n) \tag{2.54}$$

due to the additivity of entropy for product distributions, and thus we can alternatively interpret $H(X)$ as the entropy the source creates per symbol. (The latter interpretation is especially useful when we want to generalise the concept of typical sets beyond memoryless sources.)

The properties of the typical set mentioned above can now be formalised as follows:

Proposition 2.21 (Asymptotic equipartition property)**.** *Let $\epsilon \in (0,1)$. The sequence of typical sets $A_\epsilon^{(n)}(\boldsymbol{X})$ for $n \in \mathbb{N}$ has the following properties:*

1. *$H(X) - \epsilon \leq \frac{1}{n} \log \frac{1}{P_{X^n}(x^n)} \leq H(X) + \epsilon$ for all sequences $x^n \in A_\epsilon^{(n)}(\boldsymbol{X})$ and $n \in \mathbb{N}$.*
2. *$\lim_{n\to\infty} P\big[X^n \in A_\epsilon^{(n)}(\boldsymbol{X})\big] = 1$.*
3. *For all $n \in \mathbb{N}$, the size of the set satisfies*

$$|A_\epsilon^{(n)}(X)| \leq 2^{n(H(X)+\epsilon)}\,. \tag{2.55}$$

The name *asymptotic equipartition property* alludes to the the first (and defining) property of the typical set, which ensures that all sequences in the set are approximately equally likely. More precisely, the definition implies that we have

$$\left| \frac{P_{X^n}(x^n)}{P_{X^n}(\tilde{x}^n)} \right| \leq \exp(2n\epsilon) \tag{2.56}$$

for all typical sequences $x^n, \tilde{x}^n \in A_\epsilon^{(n)}(\boldsymbol{X})$.

Proof. The first property is immediate from the definition (and holds for all n).

The second property follows from the weak law of large numbers. To see this, we consider the random variables X_i produced by the source and the new random variables $Z_i = \log \frac{1}{P_X(X_i)} - H(X)$.

Question 2.22. *Verify that Z_i has zero mean and that the sequence $(Z_1, Z_2, \dots, Z_n)$ is i.i.d..*

We would now like to express the probability $P\big[X^n \in A_\epsilon^{(n)}(\boldsymbol{X})\big]$ in terms of the random variables Z_i we just introduced. We find the following sequence of equalities:

$$P\big[X^n \in A_\epsilon^{(n)}(\boldsymbol{X})\big] = P\left[\left|\frac{1}{n}\log\frac{1}{P_{X^n}(X^n)} - H(X)\right| \le \epsilon\right] \tag{2.57}$$

$$= P\left[\left|\frac{1}{n}\log\frac{1}{\prod_{i=1}^n P_X(X_i)} - H(X)\right| \le \epsilon\right] \tag{2.58}$$

$$= P\left[\left|\frac{1}{n}\sum_{i=1}^n\left(\log\frac{1}{P_X(X_i)} - H(X)\right)\right| \le \epsilon\right] \tag{2.59}$$

$$= P\left[\left|\frac{1}{n}\sum_{i=1}^n Z_i\right| \le \epsilon\right] \tag{2.60}$$

$$= 1 - P\left[\left|\frac{1}{n}\sum_{i=1}^n Z_i\right| > \epsilon\right] \tag{2.61}$$

Now since Z_i are i.i.d. and zero mean we can apply the weak law of large numbers (Proposition 0.22), which ensures that

$$\lim_{n\to\infty} P\left[\left|\frac{1}{n}\sum_{i=1}^n Z_i\right| > \epsilon\right] = 0\,, \tag{2.62}$$

and so, in particular, we have $\lim_{n\to\infty} P\big[X^n \in A_\epsilon^{(n)}(\boldsymbol{X})\big] = 1$.

The third property follows by a basic counting argument. Since every sequence in $A_\epsilon^{(n)}(\boldsymbol{X})$ has probability at least $2^{-n(H(X)+\epsilon)}$ by definition, there can be at most $2^{n(H(X)+\epsilon)}$ such sequences in the typical set as otherwise the total probability of all sequences would exceed 1. □

Proof of achievability of Theorem 2.18. We fix $\epsilon \in (0,1)$ for the moment and construct encoders and decoders for each blocklength n, working at rate $R = H(X) + 2\epsilon$. The main idea is to do a faithful encoding of all the sequences x^n in the typical set and essentially ignore and accept errors for sequences that are not typical.

To do this we index the elements of $A_\epsilon^{(n)}(X)$ in some order (say lexicographic). This simply means that to each sequence $x^n \in A_\epsilon^{(n)}(X)$, we assign a unique index $\mathrm{idx}(x^n) \in \{0,1\}^{L_n}$. By the upper bound on the size of the typical set, we know that the codeword length can be bounded as $L_n \le \lceil n(H(X)+\epsilon)\rceil \le n(H(X)+\epsilon)+1 \le nR-1 \le \lfloor nR\rfloor$, where the last inequality holds for sufficiently large n such that $\epsilon \ge \frac{2}{n}$. The assignment function is known to both the encoder and the decoder.

We now design the encoder and decoder for blocklength n.

Encoder e_n: If the realised sequence is typical, i.e. $X^n \in A_\epsilon^{(n)}(X)$, then output the index $M = \mathrm{idx}(X^n)$. Otherwise set $M = 0^{L_n}$. In other words, the precise working of the encoder is

$$e_n(x^n) = \begin{cases} \mathrm{idx}(x^n) & x^n \in A_\epsilon^{(n)}(X) \\ 0^{L_n} & x^n \notin A_\epsilon^{(n)}(X) \end{cases} \tag{2.63}$$

Decoder: Given m, output x^n such that $m = \mathrm{idx}(x^n)$. This choice is unique if such an x^n exists. If no such x^n exists, we may output any sequence of source symbols. In other words, look in the table for the sequence that corresponds to the index m.

We can now compute the probability of error for this encoder and decoder. Suppose first that the realised sequence is typical. We will never make an error because the sequence that is output coincides with the emitted sequence of the DMS, by construction of the encoder and decoder. Thus, we can only make an error if the emitted source sequence is atypical. Hence,

$$\gamma_n = P(\hat{X}^n \neq X^n) \leq P(X^n \notin A_\epsilon^{(n)}(X)) \,. \tag{2.64}$$

and in particular $\lim_{n\to\infty} \gamma_n = 0$. This implies that the sequence of codeword lengths L_n is achievable. We can thus conclude that the rate $H(X) + 2\epsilon$ is achievable. Since $\epsilon > 0$ is arbitrarily small, we have

$$R^*(\boldsymbol{X}) = \inf\{R : R \text{ is achievable}\} \leq H(X) \,. \tag{2.65}$$

□

2.3.4 Strong converse via typical sets

We can also argue with typical sets to get a stronger statement for our converse bound. In

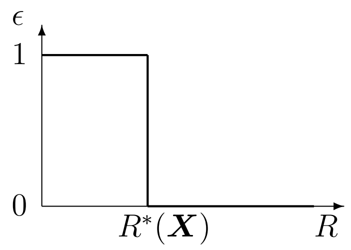


Figure 2.4: Illustration of the strong converse property. For any rate sequence of $(n, 2^{nR})$ codes with rate $R < R^*(\boldsymbol{X}) = H(X)$, the asymptotic error $\lim_{n\to\infty} P(\hat{X}^n \neq X^n)$ necessarily converges to 1.

fact, the converse via Fano's inequality is quite conservative. Even if we allow that

$$\limsup_{n\to\infty} P(\hat{X}^n \neq X^n) \leq \epsilon \tag{2.66}$$

for any $\epsilon \in [0, 1)$, it turns out that the ϵ-optimal rate must be no smaller than $H(X)$. We state this formally as follows:

Theorem 2.23. *For any sequence of $(n, 2^{\lfloor nR \rfloor})$-codes with $R < H(X)$, it necessarily holds that $P(\hat{X}^n \neq X^n) \to 1$ as $n \to \infty$.*

This theorem removes all hope to devise a more efficient source coding scheme that can beat the compression rate $H(X)$ by allowing some small error.

One way to prove this statement is to expand our characterisation of the typical set.

Proposition 2.24 (Asymptotic equipartition property, continued)**.** *Let $\epsilon, \mu \in (0,1)$. Then there exists an N_0 such that for all $n \geq N_0$, the following holds:*

1. $P\big[X^n \in A_\epsilon^{(n)}(\boldsymbol{X})\big] \geq 1 - \mu$.
2. *The size of the set satisfies*

$$|A_\epsilon^{(n)}(X)| \geq (1-\mu) 2^{n(H(X)-\epsilon)} \,. \tag{2.67}$$

Proof. The first statement is a simple consequence of the second property in Proposition 2.21, i.e. since $\lim_{n\to\infty} P\big[X^n \in A_\epsilon^{(n)}(\boldsymbol{X})\big] = 1$ there must exist an N_0 with the desired property by definition of the limit. The second statement now again follows by a counting argument. Since we know that every sequence $x^n \in A_\epsilon^{(n)}(X)$ satsifies $P_{X^n}(x^n) \leq 2^{-n(H(X)-\epsilon))}$ we will need at least $(1-\mu)2^{n(H(X)-\epsilon)}$ such elements to reach a total probability of $1-\mu$ as stipulated by the first property above. □

This allows us to lay out the idea for a proof of Theorem 2.23. Let us assume that $R < H(X)$, and define $\epsilon = \frac{1}{2}\big(H(X) - R\big)$. Then using the bound in Eq. (2.67), we find

$$2^{nR} = 2^{n(H(X)-2\epsilon)} \leq \frac{2^{-n\epsilon}}{1-\mu} |A_\epsilon^{(n)}(X)| \,. \tag{2.68}$$

For sufficiently large n this implies that $|M| < |A_\epsilon^{(n)}(X)|$, and in fact $|M|$ is smaller by a factor that grows exponentially in n. This implies that we can only faithfully represent a smaller and smaller fraction of all typical source sequences in M. Moreover, since all typical sequences are almost equiprobable this induces an error approaching 1 exponentially fast.

We finally give a more formal proof of Theorem 2.23 that uses the structure of encoder and decoder more explicitly.

Proof of Theorem 2.23. We assume $R < H(X)$ and try to give a lower bound on the probability of error.

Fix an arbitrary encoder $e_n : \mathcal{X}^n \to \{1, 2, \dots, 2^{nR}\}$. This encoder induces a partition of the space $\mathcal{X}^n$ into disjoint subsets $\mathcal{D}_m \subset \mathcal{X}^n, m \in \{1, 2, \dots, 2^{nR}\}$ defined as $\mathcal{D}_m = \{x^n : e_n(x^n) = m\}$. The best decoder (the one that minimizes the error probability) is the one that uses the following rule given message m:

$$d_n(m) = \operatorname*{argmax}_{x^n \in \mathcal{D}_m} P_{X^n}(x^n). \tag{2.69}$$

This is known as the maximum likelihood decoder as the decoder maximizes the likelihood of the observed data. We also denote the resulting random variable by $\hat{X}^n = d_n(e_n(X^n))$ as usual. Fix now $\epsilon \in (0,1)$ small enough such that $R < H(X) - 2\epsilon$ and also fix $\mu \in (0,1)$.

Question 2.25. *Argue why such an $\epsilon > 0$ always exists.*

The error probability can be bounded as

$$\begin{aligned}
\epsilon_n &= 1 - P(\hat{X}^n = X^n) && (2.70)\\
&= 1 - \sum_{m=1}^{2^{nR}} \sum_{x^n \in \mathcal{D}_m} P_{X^n}(x^n) P(\hat{X}^n = X^n | X^n = x^n) && (2.71)\\
&= 1 - \sum_{m=1}^{2^{nR}} P_{X^n}(d_n(m)) && (2.72)\\
&= 1 - \sum_{m: d_n(m) \in \mathcal{A}_\epsilon^{(n)}(X)} P_{X^n}(d_n(m)) - \sum_{m: d_n(m) \notin \mathcal{A}_\epsilon^{(n)}(X)} P_{X^n}(d_n(m)) && (2.73)\\
&\overset{(a)}{\geq} 1 - \sum_{m: d_n(m) \in \mathcal{A}_\epsilon^{(n)}(X)} P_{X^n}(d_n(m)) - \mu && (2.74)\\
&\overset{(b)}{\geq} 1 - \sum_{m: d_n(m) \in \mathcal{A}_\epsilon^{(n)}(X)} 2^{-n(H(X)-\epsilon)} - \mu && (2.75)\\
&\geq 1 - \sum_{m=1}^{2^{nR}} 2^{-n(H(X)-\epsilon)} - \mu && (2.76)\\
&\overset{(c)}{=} 1 - 2^{-n(H(X)-R-\epsilon)} - \mu && (2.77)
\end{aligned}$$

where (a) follows because the probability of the atypical set is smaller than μ for sufficiently large n according to Proposition 2.24, (b) follows since the probability of sequences in the typical set is at most $2^{-n(H(X)-\epsilon)}$ and (c) because the number of messages is 2^{nR}. Hence, since $R < H(X) - 2\epsilon$, we find

$$\liminf_{n\to\infty} \epsilon_n \geq \liminf_{n\to\infty} 1 - \mu - 2^{-n\epsilon} = 1 - \mu \,. \tag{2.78}$$

Since μ can be arbitrarily small, we in fact have $\lim_{n\to\infty} P_e^{(n)} = 1$, concluding the proof. $\square$

Chapter 3

Cryptography: Randomness extraction

[Week 6]

Intended learning outcomes:

- You can compute and use guessing probability and min-entropy.
- You can construct a randomness extractor using a family of hash functions.
- You understand that deterministic functions cannot increase entropy.

3.1 Problem setup

One of the most prominent concepts in cryptography is randomness, and it lies at the core of information-theoretic security. To understand, for example, whether a given bit string is *random*, we do not want to look at a particular instance of the string (although that is interesting as well and leads ultimately to the notion of algorithmic randomness) but instead want to check that the process that created the bit string selected it at random. Similarly and maybe even more evidently, the concept of a *secret* bit string cannot be defined unless we look at the process by which a random variable is produced. If the random process is such that the bit string is independent of any side information held by an eavesdropper, then secrecy (relative to that eavesdropper) can be claimed.

In the following we say that a random variable Z on an alphabet $\mathcal{Z}$ is close to uniformly random if it pmf is close to a uniform pmf in total variation distance (tvd), i.e. if

$$\delta_{\mathrm{tvd}}(P_Z, U_Z) := \frac{1}{2}\left\|P_Z - U_Z\right\|_1 = \frac{1}{2}\sum_{z\in\mathcal{Z}} |P_Z(z) - U_Z(z)| \tag{3.1}$$

is small, where U_Z denotes the uniform distribution on $\mathcal{Z}$. In the next chapter we learn more about the total variation distance and its use in statistics.

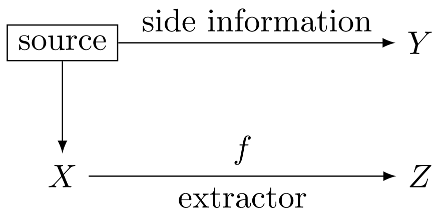


Figure 3.1: The setup of randomness extraction. A source produces random variables X and Y, where the latter variable Y is considered as side information on X. An extractor f is used to create a new random variable Z that is (close to) uniform and independent of Y. An important special case occurs when Y is trivial and we do not have any side information.

Question 3.1. *Verify that the total variation distance is a metric: it is symmetric, it is positive and zero only if the two distributions are equal, and it satisfies the triangle inequality.*

The total variation distance satisfies the data-processing inequality, i.e. for any channel $W_{Y|X}$ and any two pmfs P_X and Q_X, we have

$$\delta_{\text{tvd}}(P_X, Q_X) \geq \delta_{\text{tvd}}(P_Y, Q_Y), \tag{3.2}$$

where $P_Y(y) = \sum_x P_X(x) W_{Y|X}(y|x)$ and $Q_Y(y) = \sum_x Q_X(x) W_{Y|X}(y|x)$. This can be understood as saying that after we apply a channel $W_{Y|X}$, that is, introduce some noise, the output distributions are generally closer than the input distributions. So in a sense the two distributions have become more difficult to distinguish after applying the channel. We will verify this property in the homework.

We can now extend the definition of uniformity to the case where some side information Y on Z is available, and we want to make sure that the randomness is in fact not only uniform but also independent of the side information. This leads us to the following more general definition.

Definition 3.2. *Let P_{ZY} be a joint pmf for two random variables Z on $\mathcal{Z}$ and Y on $\mathcal{Y}$. For any $\epsilon \in (0,1)$, we say that Z is ϵ-uniformly random and independent of Y if*

$$\delta_{tvd}(P_{ZY}, U_Z \times P_Y) \leq \epsilon\,. \tag{3.3}$$

We will now consider the task of randomness extraction, namely the task of creating approximately uniform and independent random variables from a random source X that is generally neither uniform nor independent of Y. In cryptography the i.i.d. assumption (as appears for example in memoryless sources) is often not very natural since we often cannot guarantee that a random source is exactly memoryless and thus we want to ensure that our extraction scheme works even if we do not make any assumptions on the structure of the source. See Figure 3.1 for a schematic.

This is generally difficult: one thing we can immediately notice is that if one output of the source is very likely, for example if it appears with probability 0.5, then we can produce exactly one bit of perfect randomness from this source (the new uniform random variable would be the indicator function for this event, which takes the value 0 and 1 with probability 0.5 each.), and this is in fact the best we can hope for. The maximal probability over any output of the source thus appears prominently in the analysis of randomness extraction, even in the approximate case, and we will introduce it formally in the next section in terms of guessing probability and min-entropy.

Let us now formally define a randomness extractor for a fixed source, which takes X and produces a bit string Z that is uniformly random and independent of Y.

Definition 3.3. *An (ϵ, ℓ)-extractor for a source X with side information Y governed by a pmf P_{XY} is a function $f : \mathcal{X} \to \{0,1\}^\ell$ such that*

$$\delta_{tvd}(P_{ZY}, U_Z \times P_Y) \leq \epsilon \qquad \textit{where} \qquad Z = f(X) \tag{3.4}$$

and thus P_{ZY} is the distribution induced by f, i.e. $P_{ZY}(z,y) = \sum_{x: f(x)=z} P_{XY}(x,y)$.

Question 3.4. *Why should we not allow random functions/channels as extractors here?*

We may now ask for the maximum length ℓ of such an approximately uniform and independent string of bits. For this purpose we define

$$\ell_\epsilon^*(X|Y)_P := \max\left\{\ell \in \mathbb{N} : \exists \text{ an } (\epsilon, \ell)\text{-extractor for } P_{XY}\right\}. \tag{3.5}$$

We will now find bounds on this quantity from above and below that hold for arbitrary distributions P_{XY}. These bounds will be in terms of the smooth min-entropy of the source, which we will introduce in the next section. In the homework we will also consider the special case where these sources are memoryless.

3.2 Guessing probability and min-entropy

In this section we consider a source with side information given by a joint pmf P_{XY} on two random variables X on $\mathcal{X}$ and Y on $\mathcal{Y}$. We characterise our source using the concepts of guessing probability and min-entropy. The guessing probability of X given Y is the probability that an observer with access to Y can correctly guess the value of X. It is not difficult to find the optimal strategy for this task: given a sample $y \in \mathcal{Y}$, the observer will simply choose its guess as

$$\hat{x} = \operatorname*{argmax}_{x \in \mathcal{X}} P_{X|Y}(x|y). \tag{3.6}$$

The average probability of guessing the correct value of X is thus given by the guessing probability as defined in the following.

Definition 3.5. *Let P_{XY} be a joint pmf as above. The* guessing probability *of X conditioned on Y is defined as*

$$p_{\text{guess}}(X|Y)_P := \sum_{y \in \mathcal{Y}} P_Y(y) \max_{x \in \mathcal{X}} P_{X|Y}(x|y). \tag{3.7}$$

Moreover, the conditional min-entropy *of X conditioned on Y is defined as*

$$H_{\min}(X|Y)_P := -\log p_{\text{guess}}(X|Y)_P\,. \tag{3.8}$$

Example 3.6. *Consider a source with joint probability distribution on alphabets $\mathcal{X} = [4]$ and $\mathcal{Y} = [2]$ with P_Y and P_X uniform and $P_{X|Y}(\cdot|1) = (\frac{1}{2}, \frac{1}{4}, \frac{1}{4}, 0)$ and $P_{X|Y}(\cdot|2) = (0, \frac{1}{4}, \frac{1}{4}, \frac{1}{2})$. In this case, $p_{\text{guess}}(X|Y)_P = \frac{1}{2}$ since the best guesses, $\hat{x} = 1$ for $y = 1$ or $\hat{x} = 4$ for $y = 2$, respectively, are correct with probability $\frac{1}{2}$ in each case. As we seee below, we will thus only be able to extract at most $H_{\min}(X|Y)_P = 1$ perfect bit of secret randomness. This can be done by choosing an extractor $f(x) = 1$ if $x \in \{1,4\}$ and $f(x) = 2$ if $x \in \{2,3\}$. Then, for $Z = f(X)$ we find $P_{Z|Y}(\cdot|1) = P_{Z|Y}(\cdot|2) = (\frac{1}{2}, \frac{1}{2})$. Hence, Z is not correlated to Y.*

The min-entropy belongs to a class of Rényi entropies that have found widespread use in information theory, and we will explore that connection in the homework. For now let us just point out that it is always smaller than the Shannon entropy.

Lemma 3.7. *For any joint pmf P_{XY}, we have $H_{\min}(X|Y) \leq H(X|Y)$.*

Proof. To see this, we use Jensen's inequality on the convex function $t \mapsto -\log t$ to find

$$\begin{aligned} H_{\min}(X|Y) &= -\log \left(\sum_{y \in \mathcal{Y}} P_Y(y) \max_{x \in \mathcal{X}} P_{X|Y}(x|y) \right) & (3.9)\\ &\leq \sum_{y \in \mathcal{Y}} P_Y(y) \min_{x \in \mathcal{X}} \Big(-\log P_{X|Y}(x|y) \Big) & (3.10)\\ &\leq \sum_{y \in \mathcal{Y}} P_Y(y) \sum_{x \in \mathcal{X}} P_{X|Y}(x|y) \Big(-\log P_{X|Y}(x|y) \Big) = H(X|Y)\,, & (3.11) \end{aligned}$$

where for the second inequality we used the fact that the minimum over x is smaller than the expectation over x under the distribution $P_{X|Y}(x|y)$. ☐

We will state our results using a variation of the min-entropy, the *smooth min-entropy*, which is the maximum of the min-entropy over a set of distributions that are close to P_{XY} in total variation distance.

Definition 3.8. *Let P_{XY} a joint pmf and $\epsilon \in [0,1)$. We define the ϵ-smooth min-entropy of X conditioned on Y as*

$$H_{\min}^{\epsilon}(X|Y)_P := \max \Big\{ H_{\min}(X|Y)_{\tilde{P}} : \delta_{\text{tvd}}(\tilde{P}_{XY}, P_{XY}) \leq \epsilon \Big\}\,. \tag{3.12}$$

We can relate the smooth entropy to the Shannon entropy again if we consider a memoryless source $(\boldsymbol{X}, \boldsymbol{Y})$ producing sequences X^n and Y^n. In that case we have the following relation, which we will not prove here:

$$\forall \epsilon \in (0,1): \quad \lim_{n \to \infty} \frac{1}{n} H_{\min}^{\epsilon}(X^n|Y^n) = H(X|Y)\,. \tag{3.13}$$

3.3 Achievability via two-universal hash functions

There are several ways to construct extractors, including using the property of typical sets that all its elements are approximately equally likely. Here we follow a different approach (which is quite standard in cryptography) and use a random construction based on hash functions. In particular, we consider a family of two-universal hash functions $\{f_s\}_{s\in\mathcal{S}}$ where $f_s : \mathcal{X} \to \{0,1\}^\ell$. They are parametrised by a seed s and have the property that

$$\sum_{s\in\mathcal{S}} \frac{1}{|\mathcal{S}|} 1\{f_s(x) = f_s(x')\} \le \frac{1}{2^\ell} \qquad \forall\, x \neq x' \,. \tag{3.14}$$

This is the behaviour we expect from a function that produces completely random output. Such families of hash functions exist if we choose $\mathcal{S}$ large enough, but constructing them is out of the scope of this lecture.

Question 3.9. *Can you nonetheless come up with such a family for the case where $X, Z \in \{0,1\}^\ell$ as well? Your knowledge of finite fields might be helpful!*

Let us know apply a function f_S from a two-universal family of hash functions for $S \in \mathcal{S}$ chosen uniformly at random to get an output $Z = f_S(X)$.

Theorem 3.10. *Let $\{f_s\}_s$ be a two-universal family of hash functions. Using the notation introduced above, we have*

$$\sum_{s\in\mathcal{S}} \frac{1}{|\mathcal{S}|} \delta_{\text{tvd}}(P^s_{ZY}, U_Z \times P_Y) \le \frac{1}{2}\sqrt{2^{\ell - H_{\min}(X|Y)}}, \tag{3.15}$$

where $P^s_{ZY}(z,y) = \sum_{x\in\mathcal{X}:f_s(x)=z} P_{XY}(x,y)$.

Proof. Without loss of generality we can assume that the marginal P_Y has full support as otherwise we can just remove unused symbols. Using the Cauchy-Schwarz inequality in Lemma 0.24, we can bound

$$\begin{aligned}
2\delta_{\text{tvd}}(P^s_{ZY}, U_Z \times P_Y) &= \|P^s_{ZY} - U_Z \times P_Y\|_1 && (3.16)\\
&= \left\| \left(1_Z \times P_Y^{\frac{1}{2}}\right) \cdot \left(1_Z \times P_Y^{-\frac{1}{2}}\right) \cdot (P^s_{ZY} - U_Z \times P_Y) \right\|_1 && (3.17)\\
&\le \left\| 1_Z \times P_Y^{\frac{1}{2}} \right\|_2 \left\| \left(1_Z \times P_Y^{-\frac{1}{2}}\right) \cdot (P^s_{ZY} - U_Z \times P_Y) \right\|_2 && (3.18)\\
&= \sqrt{\underbrace{\sum_{z,y} P_Y(y)}_{2^\ell}} \sqrt{\sum_{z,y} P_Y^{-1}(y) \underbrace{\left(P^s_{ZY}(z,y) - U_Z(z)P_Y(y)\right)^2}_{P^s_{ZY}(z,y)^2 - 2\cdot 2^{-\ell} P^s_{ZY}(z,y)P_Y(y) + 2^{-2\ell}P_Y(y)^2}} && (3.19)\\
&= \sqrt{2^\ell \sum_{z,y} P_Y^{-1}(y) P^s_{ZY}(z,y)^2 - 2\cdot 2^{-\ell} P^s_{ZY}(z,y) + 2^{-2\ell} P_Y(y)} && (3.20)\\
&= \sqrt{\sum_{z,y} 2^\ell P_Y^{-1}(y) P^s_{ZY}(z,y)^2 - 1}\,, && (3.21)
\end{aligned}$$

where we introduced the vector 1_Z for which $1_Z(z) = 1$ for all $z \in \mathcal{Z}$, and recall that $U_Z(z) = 2^{-\ell}$ is the uniform distribution on $\mathcal{Z}$.

Using Jensen's inequality, the expectation over the seed S of the above quantity can then be bounded as

$$\sum_{s \in \mathcal{S}} \frac{1}{|\mathcal{S}|} \delta_{\text{tvd}}(P^s_{ZY}, U_Z \times P_Y) \leq \frac{1}{2} \sqrt{2^\ell \sum_{s \in \mathcal{S}} \frac{1}{|\mathcal{S}|} \sum_{z,y} P_Y^{-1}(y) P^s_{ZY}(z,y)^2 - 1} \tag{3.22}$$

It remains to analyse the term

$$\begin{aligned}
&\sum_{s \in \mathcal{S}} \frac{2^\ell}{|\mathcal{S}|} \sum_{z,y} P_Y^{-1}(y) P^s_{ZY}(z,y)^2 && (3.23)\\
&= \sum_{s \in \mathcal{S}} \frac{2^\ell}{|\mathcal{S}|} \sum_y P_Y(y) \sum_z \sum_{x,x'} 1\{f_s(x) = z\} 1\{f_s(x') = z\} P_{X|Y}(x|y) P_{X|Y}(x'|y) && (3.24)\\
&= \sum_{s \in \mathcal{S}} \frac{2^\ell}{|\mathcal{S}|} \sum_y P_Y(y) \sum_{x,x'} 1\{f_s(x) = f_s(x')\} P_{X|Y}(x|y) P_{X|Y}(x'|y) && (3.25)
\end{aligned}$$

We may now treat the cases where $x \neq x'$ and where $x = x'$ distinctly. In the first case we can apply the property of two-universal hash functions in (3.14). This yields the following bound:

$$\begin{aligned}
&\sum_{s \in \mathcal{S}} \frac{2^\ell}{|\mathcal{S}|} \sum_{z,y} P_Y^{-1}(y) P^s_{ZY}(z,y)^2 && (3.26)\\
&\leq \sum_y P_Y(y) \sum_{x \neq x'} P_{X|Y}(x'|y) P_{X|Y}(x'|y) + 2^\ell \sum_{x,y} P_Y(y) P_{X|Y}(x|y)^2 && (3.27)\\
&\leq 1 + 2^\ell \sum_y P_Y(y) \max_x P_{X|Y}(x|y) && (3.28)\\
&= 1 + 2^\ell p_{\text{guess}}(X|Y)\,. && (3.29)
\end{aligned}$$

Hence, plugging this into Eq. (3.21), we arrive at the desired bound. □

We then arrive at the following result.

Theorem 3.11. *Consider $\epsilon \in (0,1)$ and a source with pmf P_{XY}. As long as*

$$\ell \leq H^{\frac{\epsilon}{4}}_{\min}(X|Y) - 2\log\frac{1}{\epsilon}, \tag{3.30}$$

there exists an (ϵ, ℓ)-extractor for P_{XY}. This implies that

$$\ell^*_\epsilon(X|Y)_P \geq H^{\frac{\epsilon}{4}}_{\min}(X|Y) - 2\log\frac{1}{\epsilon} - 1\,. \tag{3.31}$$

Proof. Let $\tilde{P}_{XY}$ denote the distribution that achieves the maximum for the smooth min-entropy, i.e. $H_{\min}^{\frac{\epsilon}{4}}(X|Y)_P = H_{\min}(X|Y)_{\tilde{P}}$. Theorem 3.10 applied for the source $\tilde{P}_{XY}$ with the above choice of ℓ yields

$$\sum_{s\in\mathcal{S}} \frac{1}{|\mathcal{S}|}\delta_{\text{tvd}}(\tilde{P}^s_{ZY}, U_Z \times \tilde{P}_Y) \le \frac{\epsilon}{2}\,. \tag{3.32}$$

Hence, there is at least one seed value s for which this bound holds, and it remains to show that f_s constitutes an (ϵ,ℓ)-extractor. However, $\delta_{\text{tvd}}(\tilde{P}_{XY}, P_{XY}) \le \frac{\epsilon}{4}$ implies $\delta_{\text{tvd}}(\tilde{P}_Y, P_Y) \le \frac{\epsilon}{4}$ and $\delta_{\text{tvd}}(\tilde{P}^s_{ZY}, P^s_{ZY}) \le \frac{\epsilon}{4}$ by the data-processing inequality. And hence, using the triangle inequality twice we have

$$\begin{aligned} &\delta_{\text{tvd}}(P^s_{ZY}, U_Z \times P_Y) && (3.33)\\ &\quad \le \delta_{\text{tvd}}(\tilde{P}^s_{ZY}, P^s_{ZY}) + \delta_{\text{tvd}}(\tilde{P}^s_{ZY}, U_Z \times \tilde{P}_Y) + \delta_{\text{tvd}}(U_Z \times \tilde{P}_Y, U_Z \times P_Y) && (3.34)\\ &\quad \le \epsilon\,. && (3.35)\end{aligned}$$

□

3.4 Converse via an entropy inequality

The converse relies on a generalization of the following lemma, which states that applying a function to a random variable cannot increase the uncertainty about it.

Lemma 3.12. *Let $f : \mathcal{X} \to \mathcal{Z}$ be a function. Then $H(X) \ge H(f(X))$ and $H_{\min}(X) \ge H_{\min}(f(X))$.*

Proof. Let $Z = f(X)$. The joint distribution $P_{XZ}(x,z) = P_X(x)\,1\{f(x) = z\}$ satisfies

$$H(XZ) = \sum_{x,z} P_{XZ}(x,z) \log \frac{1}{P_{XZ}(x,z)} = \sum_{x} P_X(x) \log \frac{1}{P_X(x)} = H(X)\,. \tag{3.36}$$

Hence, we can conclude that $H(X) = H(XZ) = H(Z) + H(X|Z) \ge H(Z)$.

The proof for the min-entropy cannot rely on the chain rule but by inspecting the definition of the respective guessing probabilities,

$$p_{\text{guess}}(X) = \max_x P_X(x) \quad \text{and} \quad p_{\text{guess}}(Z) = \max_z P_Z(z) = \max_z \sum_{x:f(x)=z} P_X(x), \tag{3.37}$$

we see that the second term is always at least as large as the first one, i.e. $p_{\text{guess}}(Z) \ge p_{\text{guess}}(X)$. This coincides with out intuition that the input of a function is at least as hard to guess as its output, since once you guessed the input you can get the output by just applying the function. The relation for the min-entropy then follows. □

It is really important that in the statement we only allow for deterministic functions, as otherwise the equality in Eq. (3.36) would not hold.

Question 3.13. *Give an example where the inequality is violated by a probabilistic function.*

For our argument we need something similar to the above lemma, but for smooth min-entropy and with side information. Here again we can intuitively argue that it is at least as difficult to guess the input of a function as it is to guess the output (with equality if the function is injective). Formally, we can show the following:

Lemma 3.14. *Let $\epsilon \in [0,1)$ and $f : \mathcal{X} \to \mathcal{Z}$ be a function. Then, $H_{\min}^{\epsilon}(X|Y) \geq H_{\min}^{\epsilon}(f(X)|Y)$.*

In the proof we will make the assumption that f is surjective. This can be avoided, but since it is not really restrictive we made it here to allow for a streamlined presentation.

Proof. The function f can be interpreted as a channel, $W_{Z|X}(z|x) = \delta_{z,f(x)}$. We can define an inverse channel $\widetilde{W}_{X|ZY}$ that recovers the distribution P_{XY} by Bayes' rule:

$$\widetilde{W}_{X|ZY}(x|z,y) = \frac{P_{XZ|Y}(x,z|y)}{P_{Z|Y}(z|y)} = \frac{\delta_{z,f(x)} P_{X|Y}(x|y)}{\sum_{x':f(x')=z} P_{X|Y}(x'|y)} \tag{3.38}$$

Question 3.15. *Can you see what goes wrong here if the function is not surjective?*

Since this channel only maps z to values of x with $f(x) = z$ it is in fact a proper right-inverse of $W_{Z|X}$ in the following sense. For any pdf Q_{ZY} on the output we define $\widetilde{Q}_{ZY}$ as the distribution resulting from first applying $\widetilde{W}_{X|YZ}$ and then $W_{Z|X}$ to Q_{ZY}. We then find that for all z, y,

$$\widetilde{Q}_{ZY}(z,y) = \sum_{x'} W_{Z|X}(z|x') \sum_{z'} \widetilde{W}_{X|ZY}(x'|z',y) Q_{ZY}(z',y) \tag{3.39}$$

$$= \frac{\sum_{x',z'} \delta_{z,f(x')} \delta_{z',f(x')} P_{X|Y}(x'|y) Q_{ZY}(z',y)}{\sum_{x':f(x')=z} P_{X|Y}(x'|y)} = Q_{ZY}(z,y)\,. \tag{3.40}$$

Now let us assume that the distribution Q_{ZY} is optimal for the smooth min-entropy $H_{\min}^{\epsilon}(Z|Y)_P$, i.e. $H_{\min}^{\epsilon}(Z|Y)_P = H_{\min}(Z|Y)_Q$. We can then construct

$$Q_{XY}(x,y) = \sum_{z'} \widetilde{W}_{X|YZ}(x|z',y) Q_{ZY}(z',y)\,. \tag{3.41}$$

Note now that due to Eq. (3.40) the pdf $Q_{ZY}(z,y)$ is recovered by applying the function f on the register X. By the data-processing inequality for the total variation distance we have $\delta_{\text{tvd}}(Q_{XY}, P_{XY}) \leq \delta_{\text{tvd}}(Q_{ZY}, P_{ZY}) \leq \epsilon$. Hence,

$$H_{\min}^{\epsilon}(X|Y)_P \geq H_{\min}(X|Y)_Q = -\log p_{\text{guess}}(X|Y)_Q\,. \tag{3.42}$$

Now we simply use Lemma 3.12 to show that

$$p_{\text{guess}}(X|Y)_Q = \sum_y Q_Y(y)\, p_{\text{guess}}(X)_{Q^y} \tag{3.43}$$

$$\leq \sum_y Q_Y(y)\, p_{\text{guess}}(Z)_{Q^y} = p_{\text{guess}}(Z|Y)_Q\,, \tag{3.44}$$

where $Q_X^y(x) = Q_{X|Y}(x|y)$ and $Q_Z^y(z) = Q_{Z|Y}(z|y)$, respectively. Combining this with Eq. (3.42) yields the desired bound:

$$H_{\min}^{\epsilon}(X|Y)_P \geq H_{\min}(Z|Y)_Q = H_{\min}^{\epsilon}(Z|Y)_P \,. \tag{3.45}$$

□

Now we are ready to provide an upper bound on the amount of randomness that can be extracted from a source. It matches the lower bound that we derived using two-universal hash functions, and thus we know that this construction was essentially optimal.

Theorem 3.16. *Consider $\epsilon \in (0,1)$ and a source with pmf P_{XY}. Then, any (ϵ, ℓ)-extractor for P_{XY} must satisfy*

$$\ell \leq H_{\min}^{\epsilon}(X|Y)_P \tag{3.46}$$

Or, in other words, we have $\ell_{\epsilon}^{*}(X|Y)_P \leq H_{\min}^{\epsilon}(X|Y)_P$.

Proof. Let us assume there exists a function f that constitutes an (ϵ, ℓ)-extractor. We then necessarily have

$$\delta_{\text{tvd}}(P_{ZY}, U_Z \times P_Y) \leq \epsilon \tag{3.47}$$

for $Z = f(X)$, and thus

$$H_{\min}^{\epsilon}(Z|Y)_P \geq H_{\min}(Z|Y)_{U\times P} = H_{\min}(Z)_U = \ell \,, \tag{3.48}$$

where we simply evaluated the min-entropy for the distribution $U_Z \times P_Y$, which is ϵ-close to the distribution P_{ZY}. Combining this with Lemma 3.14 yields the bound $H_{\min}^{\epsilon}(X|Y)_P \geq \ell$, and since this holds for any (ϵ, ℓ)-extractor we have shown the desired statement. □

Chapter 4

Statistics: Binary hypothesis testing

[Week 7–8]

Intended learning outcomes:

- You can compute the minimal error probability in binary hypothesis testing with known priors, and understand its relationship with the total variation distance.
- You can compute the Chernoff exponent.
- You understand the setup of asymmetric binary hypothesis testing and Stein's lemma.

4.1 Binary hypothesis testing

We consider binary hypothesis testing where we try to distinguish between two models of a random process. The random process produces a sequence of random variables $\boldsymbol{X} = (X_1, X_2, \ldots)$ that are independently drawn from some (unknown) probability distribution $Q \in \mathcal{P}(\mathcal{X})$, where we take $\mathcal{X}$ to be any discrete set. Consider the hypothesis test

$$\begin{aligned} H_0 &: Q = P_0 \\ H_1 &: Q = P_1, \end{aligned} \tag{4.1}$$

where $P_0, P_1 \in \mathcal{P}(\mathcal{X})$ are two candidate probability distributions (or models) of the random process. Our goal is to deduce, from the observation of the random sequence $\boldsymbol{X}$, which of the two hypothesis is correct. H_0 is usually called the *null-hypothesis* and H_1 the *alternate hypothesis*.

A (deterministic) *test* for the sequence $X^{(n)} = (X_1, X_2, \ldots, X_n)$ is a region $\mathcal{A}_n \subset \mathcal{X}^n$. We say that the alternate hypothesis is *accepted* for this test if the observed sequence satisfies $(x_1, x_2, \ldots, x_n) \in \mathcal{A}_n$, and it is *rejected* otherwise. If the alternate hypothesis is rejected the null-hypothesis is maintained. We can then define two kinds of errors:

$$\alpha_n(\mathcal{A}_n) := P_0^n(\mathcal{A}_n) \tag{4.2}$$

$$\beta_n(\mathcal{A}_n) := 1 - P_1^n(\mathcal{A}_n) \tag{4.3}$$

The *error of the first kind* or *type-I error*, $\alpha_n(\mathcal{A}_n)$, considers the acceptance of the alternate hypothesis even if the null-hypothesis is true. The *error of the second kind* or *type-2 error*, $\beta_n(\mathcal{A}_n)$, considers the rejection of the alternate hypothesis even though it is true.

Question 4.1. *Can you formulate this problem in the general framework of probability theory as covered in Chapter 0? What is a test in this framework?*

Ideally we would like to devise a sequence of tests such that both of these errors are small, and get smaller as n increases. We could, for example, try to compute the optimal average error (assuming a *uniform prior* on the two distributions):

$$\epsilon^*_{\mathrm{sym},n}(P_0, P_1) := \frac{1}{2} \min_{\mathcal{A}_n \subset \mathcal{X}^n} \Big(\alpha_n(\mathcal{A}_n) + \beta_n(\mathcal{A}_n) \Big) . \tag{4.4}$$

Here uniform prior means that the probability we assign to the two hypotheses prior to observing the random sequence is equal, and thus $\epsilon^*_{\mathrm{sym},n}$ is indeed the probability of making a wrong decision. However, as their names indicates, often these two hypotheses are not treated on the same footing. Indeed, the question can be easily generalised to the case when the prior over the two hypothesis is not uniform.

Example 4.2. *Assume the alternate hypothesis is that a patient is suffering from COVID-19, and the null-hypothesis is that this is not the case. The error of the first kind is then a false positive and the error of the second kind is a false negative. If we devise a test distinguishing these two hypothesis we are probably more tolerant of false positives than false negatives.*

For such cases it is natural to look at an asymmetric setting, and define, for all $\epsilon \in (0, 1)$,

$$\beta^*_n(\epsilon; P_0, P_1) := \min\{\beta_n(\mathcal{A}_n) : \alpha_n(\mathcal{A}_n) \le \epsilon\}, \tag{4.5}$$

where $\mathcal{A}_n$ runs through all subsets of $\mathcal{X}^n$. Asymmetric hypothesis testing also allows us to deal with the situation when we do not know the prior probabilities. In that case the sum (or probabilistic mixture) of the two errors does not make sense and we need to look at the errors independently. We can, however, still ask the question how these two errors trade off against each other. This is done by analysing $\beta^*_n(\epsilon)$, and in particular by looking at its asymptotics for large n.

4.2 Symmetric hypothesis testing

The first natural question is to evaluate $\epsilon^*_{\mathrm{sym},n}$ for $n = 1$. With that in hand, we will then give a bound on the asymptotic error when $n \to \infty$.

4.2.1 Total variation distance

We will see that $\epsilon^*_{\mathrm{sym},n}$ can be expressed in terms of the *total variation distance* (tvd) between two pmfs P_0 and P_1, which is given by

$$\delta_{\mathrm{tvd}}(P_0, P_1) := \frac{1}{2} \sum_{x \in \mathcal{X}} \big| P_0(x) - P_1(x) \big| . \tag{4.6}$$

The total variation distance vanishes if and only if $P_0 = P_1$ and it reaches its maximum 1 when P_0 and P_1 are orthogonal, that is, when for every $x \in \mathcal{X}$ either $P_0(x) = 0$ or $P_1(x) = 0$. We can alternatively express the TVD using the following variational formulae, which motivate its name.

Lemma 4.3. *For any two pmfs P_0 and P_1, the following relations hold:*

$$\delta_{\text{tvd}}(P_0, P_1) = \max_{\mathcal{A} \subset \mathcal{X}} \left(\sum_{x \in \mathcal{A}} P_0(x) - P_1(x) \right) \tag{4.7}$$

$$= \max_{\mathcal{A} \subset \mathcal{X}} \left(\sum_{x \in \mathcal{A}} P_1(x) - P_0(x) \right). \tag{4.8}$$

Proof. To see this equivalence, first note that

$$\sum_{x \in \mathcal{X}} P_0(x) - P_1(x) = 0 \tag{4.9}$$

by normalisation, and thus, for any set $\mathcal{A} \subseteq \mathcal{X}$, we have

$$\sum_{x \in \mathcal{A}} P_0(x) - P_1(x) = \sum_{x \in \mathcal{A}^c} P_1(x) - P_0(x). \tag{4.10}$$

Specifically, for the set $\mathcal{A} = \{x \in \mathcal{X} : P_0(x) > P_1(x)\}$ that is optimal for the maximisation in Eq. (4.7), we find

$$\sum_{x \in \mathcal{A}} \big|P_0(x) - P_1(x)\big| = \sum_{x \in \mathcal{A}^c} \big|P_1(x) - P_0(x)\big| \tag{4.11}$$

and thus

$$\max_{\mathcal{A} \subset \mathcal{X}} \left(\sum_{x \in \mathcal{A}} P_0(x) - P_1(x) \right) = \sum_{x \in \mathcal{A}} \big|P_0(x) - P_1(x)\big| = \frac{1}{2} \sum_{x \in \mathcal{X}} \big|P_0(x) - P_1(x)\big|. \tag{4.12}$$

□

The total variational distance is closely related to the 1-norm, which is defined for any vectors v_0 and v_1 that do not necessary need to be normalised. It is defined as

$$\|v_0 - v_1\|_1 := \sum_{x \in \mathcal{X}} \big|v_0(x) - v_1(x)\big|. \tag{4.13}$$

Hence, in particular, $\delta_{\text{tvd}}(P_0, P_1) = \frac{1}{2}\|P_0 - P_1\|_1$. We can now state the following result for binary hypothesis testing with general priors.

Proposition 4.4. *Let H_0 (with probability P_0) and H_1 (with probability P_1) have prior probabilities p and $1-p$, respectively. The minimal probability of error for the binary hypothesis test with $n = 1$, denoted $\epsilon^*_{p,1}(P_0, P_1)$, is given by*

$$\epsilon^*_{p,1}(P_0, P_1) = \frac{1}{2}\Big(1 - \big\|pP_0 - (1-p)P_1\big\|_1\Big) . \tag{4.14}$$

In particular, if $p = 1 - p = \frac{1}{2}$, we have $\epsilon^*_{\text{sym},1} = \frac{1}{2}\left(1 - \delta_{\text{tvd}}(P_0, P_1)\right)$. This gives a clear operational interpretation for the total variation distance, which is a widely used distance measure in statistics. On the one hand, when $P_0 = P_1$ the total variation distance vanishes and the best thing we can do is a random guess. On the other hand, when P_0 and P_1 are orthogonal, then we can distinguish them perfectly and the error vanishes.

Proof. First we observe the following relations:

$$\epsilon^*_{p,1} = \min_{\mathcal{A}\subset\mathcal{X}} \big(pP_0(\mathcal{A}) + (1-p)P_1(\mathcal{A}^c)\big) \tag{4.15}$$

$$= p - \max_{\mathcal{A}\subset\mathcal{X}} \big(pP_0(\mathcal{A}^c) - (1-p)P_1(\mathcal{A}^c)\big) \tag{4.16}$$

$$= 1 - p - \max_{\mathcal{A}\subset\mathcal{X}} \big((1-p)P_1(\mathcal{A}) - pP_0(\mathcal{A})\big) \tag{4.17}$$

Combining the two relations we get

$$2\epsilon^*_{p,1} = 1 - \max_{\mathcal{A}\subset\mathcal{X}} \big((1-p)P_1(\mathcal{A}) - pP_0(\mathcal{A})\big) - \max_{\mathcal{A}\subset\mathcal{X}} \big(pP_0(\mathcal{A}^c) - (1-p)P_1(\mathcal{A}^c)\big) \tag{4.18}$$

At this point we can determine which sets achieve the maximum in these two optimisation. Clearly both are achieved by the set $\mathcal{A} = \{x \in \mathcal{X} : (1-p)P_1(x) \geq pP_0(x)\}$. The above expression then simplifies to

$$2\epsilon^*_{p,1} = 1 - \sum_{x\in\mathcal{A}} (1-p)P_1(x) - pP_0(x) - \sum_{x\in\mathcal{A}^c} pP_0(x) - (1-p)P_1(x) \tag{4.19}$$

$$= 1 - \sum_{x\in\mathcal{A}} \big|pP_0(x) - (1-p)P_1(x)\big| - \sum_{x\in\mathcal{A}^c} \big|pP_0(x) - (1-p)P_1(x)\big| \tag{4.20}$$

$$= 1 - \sum_{x\in\mathcal{X}} \big|pP_0(x) - (1-p)P_1(x)\big| \tag{4.21}$$

$$= 1 - \big\|pP_0 - (1-p)P_1\big\|_1 . \tag{4.22}$$

Dividing both sides by 2 then yields the desired result. □

4.2.2 Chernoff exponent

When we look at n i.i.d. copies of the sample, distributed according to P_0^n or P_1^n, respectively, we make the at first sight surprising observation that these two distributions get closer and closer to orthogonal as $n \to \infty$ (unless $P_0 = P_1$, of course). Or, in other words, the total variation distance between P_0^n and P_1^n converges to 1 as $n \to \infty$.

This can be verified using a variation of the typical sets we discussed in Chapter 2. We define the empirical typical set of P_0 as

$$\mathcal{A}^{(n)}_{\text{emp},\epsilon} := \{x^n \in \mathcal{X}^n : \delta_{\text{tvd}}(P_{x^n}, P_0) \leq \epsilon\} \,, \tag{4.23}$$

where P_{x^n} is the empirical distribution on $\mathcal{X}$ induced by the sequence x^n, i.e.,

$$P_{x^n}(x) = \frac{1}{n} \left|\{i \in [n] : x_i = x\}\right| \,. \tag{4.24}$$

In the homework we will show that $\lim_{n\to\infty} P\left[X^n \in \mathcal{A}^{(n)}_{\text{emp},\epsilon}\right] = 1$.

We can now can define ϵ small enough so that the two empirical typical sets, $\mathcal{A}^{(n)}_{\text{emp},\epsilon}$ for P_0 and $\mathcal{B}^{(n)}_{\text{emp},\epsilon}$ for P_1 are disjoint. This is easy to see since a sequence x^n can have its empirical distribution P_{x^n} either close to P_0 or close to P_1, but not both. Using the variational formula in Eq. (4.7) we can then write the following:

$$\begin{aligned}
\delta_{\text{tvd}}(P_0^n, P_1^n) &= \max_{\mathcal{A}^n \subset \mathcal{X}} \left(\sum_{x \in \mathcal{A}^n} P_0(x^n) - P_1(x^n) \right) && (4.25)\\
&\geq \sum_{x \in \mathcal{A}^{(n)}_{\text{emp},\epsilon}} P_0(x^n) - P_1(x^n) && (4.26)\\
&= \sum_{x \in \mathcal{A}^{(n)}_{\text{emp},\epsilon}} P_0(x^n) + \sum_{x \notin \mathcal{A}^{(n)}_{\text{emp},\epsilon}} P_1(x^n) - 1 && (4.27)\\
&\geq \sum_{x \in \mathcal{A}^{(n)}_{\text{emp},\epsilon}} P_0(x^n) + \sum_{x \in \mathcal{B}^{(n)}_{\text{emp},\epsilon}} P_1(x^n) - 1\,. && (4.28)
\end{aligned}$$

But now note that both of these sums converge to 1 as $n \to \infty$ by definition of the empirical typical sets and the claim above. Hence, the total variation distance must converge to 1 as well, and thus, as $n \to \infty$,

$$\delta_{\text{tvd}}(P_0^n, P_1^n) \to 1 \qquad \text{and} \qquad \epsilon^*_{\text{sym},n} \to 0\,. \tag{4.29}$$

Or, in other words, the symmetric error $\epsilon^*_{\text{sym},n}$ converges to zero. We are also interested how fast this convergence to zero is. We will show the following bound:

Proposition 4.5. *For any two pmfs P_0 and P_1, we have*

$$\lim_{n\to\infty} -\frac{1}{n} \log \epsilon^*_{\text{sym},n}(P_0, P_1) \geq C(P_0, P_1)\,, \tag{4.30}$$

where we introduced the Chernoff distance *or Chernoff exponent,*

$$C(P_0, P_1) := -\min_{0\leq\lambda\leq 1} \log \sum_{x\in\mathcal{X}} P_0(x)^\lambda P_1(x)^{1-\lambda}\,. \tag{4.31}$$

Note that this actually corresponds to an asymptotic upper bound on the probability of error, so it says that there exists a sequence of tests for which the error drops as $2^{-nC(P_0,P_1)}$. It turns out (but we will not show this here) that this is optimal, i.e., that equality holds in Eq. (4.30).

Proof. To show the bound on the error propbability, we argue that

$$2\epsilon^*_{\text{sym},n} = \min_{\mathcal{A}_n \subset \mathcal{X}^n} P_0^n(\mathcal{A}_n) + P_1^n(\mathcal{A}_n^c) \tag{4.32}$$

$$= \sum_{x^n \in \mathcal{X}^n} \min\{P_0^n(x^n), P_1^n(x^n)\} \tag{4.33}$$

$$\leq \sum_{x^n \in \mathcal{X}^n} P_0^n(x^n)^\lambda P_1^n(x^n)^{1-\lambda} \tag{4.34}$$

$$= \sum_{x_1 \in \mathcal{X}} \dots \sum_{x_n \in \mathcal{X}} P_0(x_1)^\lambda \dots P_0(x_n)^\lambda P_1(x_1)^{1-\lambda} \dots P_1(x_n)^{1-\lambda} \tag{4.35}$$

$$= \left(\sum_{x \in \mathcal{X}} P_0(x)^\lambda P_1(x)^{1-\lambda} \right)^n \tag{4.36}$$

We now take the logarithm on both sides and divide through n to get

$$\frac{1}{n} \log \epsilon^*_{\text{sym},n} \leq \log \sum_{x \in \mathcal{X}} P_0(x)^\lambda P_1(x)^{1-\lambda} - \frac{1}{n} \tag{4.37}$$

The last term vanishes in the limit $n \to \infty$, and thus the bound in (4.30) follows by maximising the right-hand side over all choices of $\lambda \in [0,1]$. □

Question 4.6. *What changes when we do the same analysis for $\epsilon^*_{p,n}$?*

4.3 Asymmetric hypothesis testing and the information spectrum method

For simplicity we assume that $D(P_0\|P_1) < \infty$ in the following, as otherwise by definition of the relative entropy there are some $x \in \mathcal{X}$ with $P_0(x) > 0$ but $P_1(x) = 0$, and, as we will see in the homework, it is possible to come up with tests that have $\beta_n^*(\epsilon) = 0$ for large enough n.

Under this assumption, our goal is to show that regardless of the constant upper bound ϵ on the type-I error, the type-II error behaves as

$$\beta_n^*(\epsilon) \approx 2^{-nD(P_0\|P_1)}, \tag{4.38}$$

where the approximation is up to factors that are sub-exponential in n. This means that the optimal exponential rate at which the type-II error approaches zero is determined by the relative entropy (in first order), thus giving the relative entropy $D(P_0\|P_1)$ a clear operational interpretation in statistics. Let us restate this as a theorem:

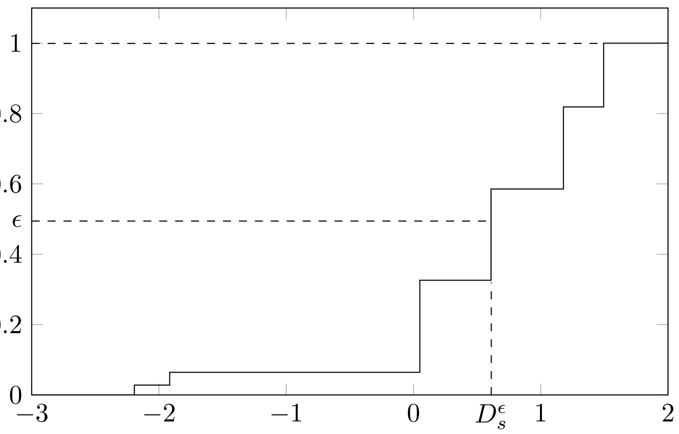


Figure 4.1: Example of the information spectrum. The plot shows the cumulative distribution of $Z = \log \frac{P(X)}{Q(X)}$ and the value of $D_s^\epsilon(P\|Q)$ for some example distributions.

Theorem 4.7 (Chernoff-Stein Lemma)**.** *For every $\epsilon \in (0, 1)$,*

$$\lim_{n\to\infty} -\frac{1}{n} \log \beta_n^*(\epsilon) = D(P_0\|P_1) \tag{4.39}$$

For the proof we will use the information spectrum method (see [**?**] for more information on that technique). Consider a random variable X that takes values on $\mathcal{X}$ and two pmfs $P_0, P_1 \in \mathcal{P}(\mathcal{X})$ as above. Recall that the log-likelihood ratio for the two pmfs is the random variable

$$Z = \log \frac{P_0(X)}{P_1(X)}, \tag{4.40}$$

where X (and thus Z) is distributed according to P_0. The log-likelihood ratio is an important random variable in the analysis of many different information processing tasks.

Question 4.8. *Verify that the expectation value of Z under P_0 is $D(P_0\|P_1)$.*

We now introduce the following expression:

$$D_s^\epsilon(P_0\|P_1) := \sup \{R \in \mathbb{R} : P_0[Z \leq R] \leq \epsilon\} \tag{4.41}$$

This quantity looks complicated at first sight, but it simply evaluates exactly where (the value R) we need to cut off the pmf for the *log-likelihood ratio*, Z, so that the probability that $Z \leq R$ is at most ϵ. One could also simply see it as an inverse of the cumulative distribution function of Z. (This result can also be interpreted as a consequence of the Neyman-Pearson lemma, which states that all tests optimising the two types of errors are threshold tests for the log-likelihood ratio.)

Lemma 4.9. *Let $n \in \mathbb{N}$, $\epsilon \in (0,1)$ and $\delta \in (0, 1-\epsilon)$. The following two inequalities hold:*

$$D_s^{\epsilon}(P_0^n \| P_1^n) \leq -\log \beta_n^*(\epsilon) \leq D_s^{\epsilon+\delta}(P_0^n \| P_1^n) + \log \frac{1}{\delta}\,. \tag{4.42}$$

For $n = 1$ this gives bounds on asymmetric hypothesis testing for any two distributions P_0 and P_1, without using the i.i.d. structure. If one plugs in n-fold i.i.d. distributions instead this recovers the result for general n. This is an example of a *one-shot bound*, a generic bound on an information-theoretic quantity that can then be easily statistically analysed by taking advantage of an i.i.d. or similar structure.

Proof. To get the lower bound, we construct a test $\mathcal{A}_{R,n} := \{x^n \in \mathcal{X}^n : P_0^n(x^n) \leq 2^R P_1^n(x^n)\}$, where $R \in \mathbb{R}$ still needs to be chosen. These tests, which simply check if the log-likelihood ratio exceeds R, are called Neyman-Pearson tests and are known to be the most powerful tests. We will actually show this optimality here implicitly, as our converse bound will match the achievability we get using this test.

Let us choose $R = D_s^{\epsilon}(P_0^n \| P_1^n) - \mu$ for some $\mu > 0$ that can be chosen arbitrarily small. The reason we need this small slack $\mu > 0$ is simply that by definition of the supremum in (4.41) this ensures that we have $\alpha_n(\mathcal{A}_{R,n}) = P_0^n(\mathcal{A}_{R,n}) \leq \epsilon$ for any $\mu > 0$, while the same might not necessarily be true at $\mu = 0$. (Recall that the supremum can be taken at the boundary of an open set.) Moreover, we have

$$\begin{aligned}
\beta_n(\mathcal{A}_{R,n}) &= P_1^n(\mathcal{A}_{R,n}^c) && (4.43)\\
&= \sum_{x^n \in \mathcal{X}^n} P_1^n(x^n)\, 1\{P_0^n(x^n) > 2^R P_1^n(x^n)\} && (4.44)\\
&\leq 2^{-R} \sum_{x^n \in \mathcal{X}^n} P_0^n(x^n)\, 1\{P_0^n(x^n) > 2^R P_1^n(x^n)\} && (4.45)\\
&\leq 2^{-R}\,. && (4.46)
\end{aligned}$$

This directly implies that $\beta_n^*(\epsilon) \leq 2^{-R}$, or, equivalently,

$$-\log \beta_n^*(\epsilon) \geq D_s^{\epsilon}(P_0^n \| P_1^n) - \mu\,. \tag{4.47}$$

Since this holds for all $\mu > 0$ we get the desired inequality.

To get the upper bound, let $\mathcal{A}_n$ be the optimal test for $\beta_n^*(\epsilon)$, i.e. we have $\alpha_n(\mathcal{A}_n) \leq \epsilon$ and $\beta_n(\mathcal{A}_n) = \beta_n^*(\epsilon)$. Recall also the definition of the log-likelihood ratio, $Z = \log \frac{P_0^n(X^n)}{P_1^n(X^n)}$. Using these properties we can establish the following sequence of inequalities:

$$
\begin{aligned}
1 - P_0^n(Z \le R) &= P_0^n\left(\log \frac{P_0^n(X^n)}{P_1^n(X^n)} > R\right) && (4.48)\\
&= \sum_{x^n \in \mathcal{X}^n} P_0^n(x^n)\, 1\{P_0^n(x^n) > 2^R P_1^n(x^n)\} && (4.49)\\
&\ge \sum_{x^n \in \mathcal{X}^n} \big(P_0^n(x^n) - 2^R P_1^n(x^n)\big)\, 1\{P_0^n(x^n) > 2^R P_1^n(x^n)\} && (4.50)\\
&\ge \sum_{x^n \in \mathcal{X}^n} \big(P_0^n(x^n) - 2^R P_1^n(x^n)\big)\, 1\{x^n \in \mathcal{A}_n^c\} && (4.51)\\
&= P_0^n(\mathcal{A}_n^c) - 2^R P_1^n(\mathcal{A}_n^c) && (4.52)\\
&= 1 - \alpha_n(\mathcal{A}_n) - 2^R \beta_n(\mathcal{A}_n) && (4.53)\\
&\ge 1 - \epsilon - 2^R \beta_n^*(\epsilon)\,. && (4.54)
\end{aligned}
$$

The critical step is to get from Eq. (4.50) to Eq. (4.51). To verify this, note that the test $1\{P_0^n(x^n) > 2^R P_1^n(x^n)\}$ is actually the one that maximises the sum since it cuts out all negative contributions. Any other test, including $\mathcal{A}_n^c$, can thus only reduce the sum.

Now, if we choose $R = \log \delta - \log \beta_n^*(\epsilon)$, the above implies that

$$
P_0^n(Z \le R) \le \epsilon + \delta \qquad (4.55)
$$

and thus we have $D_s^{\epsilon+\delta}(P_0^n \| P_1^n) \ge R \ge -\log \beta_n^*(\epsilon) + \log \delta$, which is the desired upper bound. □

In the homework you will show that the following asymptotic limit holds.[1]

Lemma 4.10. *Let $P_0, P_1 \in \mathcal{P}(X)$ such that $D(P_0\|P_1) < \infty$ and $\epsilon \in (0,1)$. Then,*

$$
\lim_{n\to\infty} \frac{1}{n} D_s^{\epsilon}(P_0^n \| P_1^n) = D(P_0\|P_1)\,. \qquad (4.56)
$$

As an aside, we can evaluate the quantity on the left, $\frac{1}{n} D_s^{\epsilon}(P_0^n \| P_1^n)$, even to higher orders in n using the central limit theorem. While we will not need this here, analysing such higher order terms has been a fruitful area of research recently as it allows us to make more precise statements about optimal errors for smaller n, and thus for practical settings where we are far from the asymptotic setting of very large n.

Proof of Theorem 4.7. The proof of the theorem is evident once we combine Lemma 4.9 and Lemma 4.10. Namely, from Lemma 4.9 we get

$$
\frac{1}{n} D_s^{\epsilon}(P_0^n \| P_1^n) \le -\frac{1}{n}\log \beta_n(\epsilon) \le \frac{1}{n} D_s^{\epsilon+\delta}(P_0^n \| P_1^n) + \frac{1}{n}\log\frac{1}{\delta} \qquad (4.57)
$$

and in the limit $n \to \infty$ both the lower and upper bound converge to the relative entropy by Lemma 4.10. □

[1]It is essentially a direct consequence of the law of large numbers applied for the random variable $Z = \sum_{i=1}^n \log P_0(X_i) - \log P_1(X_i)$, where X_i are i.i.d. distributed according to the law P_0.

Chapter 5

Error correcting codes

[Week 8–9]

Intended learning outcomes:

- You will be familiar with the concept of error correcting codes and can compute their rate and minimal distance.
- You can construct a linear code from its generator matrix or parity check matrix.
- You understand and can construct basic Reed-Solomon codes.
- [Not part of exam] You understand the basics behind low density parity check codes and the use of belief propagation for deciding them.

5.1 Definitions and bounds on codebook size

Error correcting codes are a very rich topic and are studied by communication engineers, computer scientists and mathematicians alike. In computer science, for example, they are used in complexity theory, cryptography, and the study of pseudo-randomness. We can only touch the very surface of this theory here. We will first discuss some general properties of codes and particularly linear codes, and then move on to describe one widely used class of codes in more detail, the Reed-Solomon family of codes.

In the following we will consider codewords that are strings of a fixed length, on some alphabet Σ. The following notions are useful.

Definition 5.1. *The* Hamming weight *of a string Σ^n is defined as $\big|\{i : x_i \neq 0\}\big|$, i.e., the number of nonzero elements of x^n. The* Hamming distance *between two strings $x^n, y^n \in \Sigma^n$ is defined as*

$$\delta(x^n, y^n) = \big|\{i : x_i \neq y_i\}\big|, \tag{5.1}$$

i.e., the number of locations where the strings differ.

We will now introduce the notion of an *error correction code*, or simply code for the remainder of this chapter.

Definition 5.2. *An error correction code C of length n over a finite alphabet Σ is a subset of Σ^n. The elements of C are called codewords, and C is sometimes also called the codebook. We will use the following properties and definitions:*

- *An error correction code is a* binary code *if $\Sigma = \{0,1\}$. (We will mostly consider binary codes in the following.)*
- *A binary code is a* linear code *if C is a subspace of $\{0,1\}^n$. This means that for any two codewords $c, c' \in C$, the bitwise XOR of c and c', denoted $c \oplus c'$, is an element of C as well. In particular, the all zero vector is in C.*
- *The size of the codebook is denoted by $|C|$.*
- *The rate of the code is defined as*
$$R(C) = \frac{\log|C|}{\log|\Sigma^n|} = \frac{\log|C|}{n\log|\Sigma|} \tag{5.2}$$
- *The* minimal distance *of a code C, denoted $d(C)$, is defined as*
$$d(C) = \min_{\substack{c,c' \in C \\ c \neq c'}} \delta(c, c') \tag{5.3}$$

Question 5.3. *Consider a binary code $C \in \{0,1\}^n$ where each codeword is constructed by adding a parity bit to a bit string of length $n-1$. Is this a linear code? What can you say about its minimum distance?*

The following relationships between minimal distance of a binary code and its use for error correction are rather immediate. Consider a binary code with minimum distance d. Such a code can be used to

- Detect up to $d-1$ bit flip errors.
- Correct up to $\lfloor \frac{d-1}{2} \rfloor$ bit flip errors.
- Correct up to $d-1$ erasures.

In the erasure model the decoder is informed which bits of the codeword are faulty.

The following bounds establish a relationship between these parameters, limiting the size of the code in terms of the other parameters.

Lemma 5.4 (Hamming bound)**.** *Let C be a binary code with block length n and distance d. Then,*
$$|C| \leq \frac{2^n}{\sum_{i=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{n}{i}} \tag{5.4}$$
In particular, when $d = 3$ we have $|C| \leq \frac{2^n}{n+1}$.

Proof. For every codeword c define its neighbourhood $N(c,r)$ as all the strings that differ from c in at most r locations. Setting $r = \lfloor \frac{d-1}{2} \rfloor$, we note that $N(c,r) \cap N(c',r) = \emptyset$ for any two distinct codewords c and c'. Moreover, we have

$$|N(c,r)| = \sum_{i=0}^{r} \binom{n}{i} . \tag{5.5}$$

Hence, we can write

$$2^n \geq \left| \bigcup_{c \in C} N(c,r) \right| = \sum_{c \in C} |N(c,r)| = |C| \sum_{i=0}^{r} \binom{n}{i} . \tag{5.6}$$

Solving this for $|C|$ yields the desired inequality. □

Note that for this bound to hold with equality the space must be exactly filled out by these neighbourhood balls. We call codes for which this is true *perfect codes.* The following bound applies to all codes, not only binary codes.

Lemma 5.5 (Singleton bound)**.** *Let C be a code with block length n and distance d on an alphabet with $|\Sigma| = q$. Then, we must have*

$$|C| \leq q^{n-d+1} . \tag{5.7}$$

Proof. First observe that there are q^n possible codewords. Let C be an arbitrary code of minimum distance d. Clearly, all codewords $c \in C$ are distinct. Moreover, if we puncture the code by deleting the first $d-1$ letters of each codeword, then all resulting codewords must still be pairwise different. The newly obtained codewords each have length $n-(d-1) = n-d+1$, and thus, there can be at most q^{n-d+1} of them. □

5.2 Linear codes

We will often be interested in binary codes, but it is important to note that these ideas can all be extended to the case where Σ is any finite field. Linear codes are defined on a field F_q, and codewords of length n are vectors in F_q^n. Since linear codes form subspaces we can express every codeword as a linear combination of a basis of codewords. We denote by k the *dimension* of the subspace, or the minimal number of codewords needed to for a basis. A linear code with a k-dimensional subspace of an n-dimensional space is referred to as a $[n,k]_q$*-code.* Furthermore, if it has minimum distance d, we call it an $[n,k,d]_q$*-code.* We usually drop the subscript q when it is clear from context, e.g. when we are discussing binary codes.

Definition 5.6. *Let C be an $[n,k]$-code. A matrix $G \in F_q^{n \times k}$ is said to be a* generator matrix *for C if its k columns span C.*

Using the generator matrix we can encode any binary string $x \in F_q^k$ into a codeword $c \in F_q^n$ by the matrix multiplication $c = Gx$. Note that a linear code admits different generator matrices, corresponding to the different choices of basis for the code as a vector space. This corresponds to different encodings of the messages into codewords, with the same fixed set of codewords.

Example 5.7. *Consider the binary repetition code for $n = 3$ comprised of the codewords* 000 *and* 111. *The generator matrix for this code is $G = (1, 1, 1)^T$.*

For linear codes we can give a bound on the codebook size — the Singleton bound simply evaluates to $k \leq n - d + 1$ or

$$d \leq n - k + 1\,. \tag{5.8}$$

There are two generic ways two characterise a subspace:

- By specifying a basis of the subspace, as we have done above using the generator matrix.
- By specifying a basis of the orthogonal subspace.

For linear codes that orthogonal subspace is spanned by vectors that are orthogonal to the linear subspace spanned by the codewords. Those vectors can be interpreted as parity checks.

Definition 5.8. *Let C be an $[n, k]$-code. A matrix $H \in F_q^{(n-k)\times n}$ is said to be a* parity check matrix *for C if $Hc = 0$ for every $c \in C$.*

Example 5.9. *The Hamming code is a binary $[7, 4, 3]$-code given by codewords of the form*

$$x_1, \quad x_2, \quad x_3, \quad x_4, \quad x_2 \oplus x_3 \oplus x_4, \quad x_1 \oplus x_3 \oplus x_4, \quad x_1 \oplus x_2 \oplus x_4\,. \tag{5.9}$$

A possible generator matrix for this code is given by

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix} \tag{5.10}$$

A possible parity check matrix is given by

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{5.11}$$

The Hamming code is a perfect code since $2^k = \frac{2^n}{n+1}$ for $k = 4$ and $n = 7$.

Definition 5.10. *The dual of a binary $[n,k]$-code C, the $[n,n-k]$-code $C^\perp$, is the space spanned by all codewords $c' \in F_q^n$ such that*

$$\sum_{i=1}^{n} c_i c_i' = 0 \tag{5.12}$$

for all $c \in C$.

From the definition we can see that $G^\perp = H^T$ and $H^\perp = G^T$. In particular, the dual of a dual code is the code itself.

5.3 Reed-Solomon codes

Reed-Solomon were first used to do error correction for the Voyager program and became really widespread in their use to protect against errors on compact discs. They are still used in two-dimensional bar codes like QR codes.

The Reed-Solomon code is actually a family of codes, where every code is characterised by three parameters: an alphabet size q, a block length n, and a message length k, with $k < n \leq q$. In this code a message $m = (m_0, m_1, \ldots, m_{k-1}) \in F_q^k$ is first mapped to a polynomial $p_m(x)$ with $x \in F_q$ of degree $k-1$ given by

$$p_m(x) = \sum_{i=0}^{k-1} m_i x^i. \tag{5.13}$$

The codeword for m is then obtained by evaluating p_m at n different points $x_i \in F_q$ for $i \in [n]$, i.e.

$$C(m) = \left(p_m(x_1), \ldots, p_m(x_n)\right). \tag{5.14}$$

This constitutes a linear code with the generator matrix $G \in F_q^{k\times n}$ given by

$$G = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ x_1 & x_2 & \ldots & x_n \\ \vdots & \vdots & & \vdots \\ x_1^{k-1} & x_2^{k-1} & \ldots & x_n^{k-1} \end{pmatrix}^T \tag{5.15}$$

The basic idea is that a polynomial of order $k-1$ is uniquely specified if we know its value at k points or more.

Lemma 5.11. *The above Reed-Solomon code is a $[n,k,n-k+1]_q$-code.*

Proof. The code is an $[n,k]_q$-code by construction. The only property we need to show here is that the minimal distance of the code is given by $d = n-k+1$.

On the one hand, we can generally write $\delta(c_1, c_2) = \delta(c_1 - c_2, 0)$ for any two codewords $c_1, c_2 \in C$, where $c_1 - c_2$ is also a codeword since C is linear. Hence, the minimal distance of the codebook is simply given by the minimal number of nonzero elements in any codeword that is not the all zero codeword. But since for any $m \neq 0$, the polynomial $p_m(x)$ is nontrivial and of order $k-1$, we know that it has at most $k-1$ roots, so we must have $d \geq n - (k-1) = n - k + 1$.

On the otherhand, we observe that by the Singleton bound we must have $d \leq n - k + 1$, therefore concluding the proof. $\square$

If we choose $q = 2^n$ we can interpret the Reed-Solomon code as a binary code. For $q = 2^2$, $n = 4$ and $k = 2$ we get the following mappings, where each symbol $\{0,\ 1,\ 2,\ 3\}$ can be interpreted as a binary sequence $\{00, 01, 10, 11\}$. The codewords are constructed by evaluating the polynomial at the points $\{0, 1, 2, 3\}$, using the multiplication and addition rules for F_4 discussed in Section 0.6 of Chapter 0. For example, we get

$$0011 = \{0, 3\} \to 3 + 0x \to \{3, 3, 3, 3\} = 11111111 \tag{5.16}$$

$$1001 = \{2, 1\} \to 1 + 2x \to \{1, 3, 2, 0\} = 01111000 \tag{5.17}$$

$$1010 = \{2, 2\} \to 2 + 2x \to \{2, 0, 1, 3\} = 10000111\,. \tag{5.18}$$

The above should be read as "initial bit string" = "written as two values in F_4 by interpreting it as binary representation" $\to$ "corresponding polynomial of degree 1" $\to$ "polynomial evaluated at $x = \{0, 1, 2, 3\}$ = "encoded bit string using binary representation".

5.4 Low density parity check (LDPC) codes

[The detailed content of this section will not be part of the exam; however, you should understand the basic principles behind LDPC codes.]

LDPC codes are linear codes with the property that the parity-check matrix H is sparse, i.e., the individual parities that need to be checked involve only a few of the message bits. The sparsity of H allows for a relatively efficient iterative decoding heuristic called *belief propagation*. It turns out that this heuristic works extremely well in practice, even though we cannot generally prove its convergence. On the other hand, finding the message with the maximum likelihood, as an optimal decoder would do, is a problem that we do not know how to solve in time polynomial in the block length n. These efficiency considerations are important since this implies that LDPC codes can be used for large block lengths, and in particular can be used to approach the capacity of a communication channel. We will cover the capacity of channels in the next lecture.

In this section, we only consider binary LDPC codes.

Example 5.12 (A regular $(3, 6)$-LDPC code)**.** *Consider a binary linear code specified by its* *parity-check matrix (n is even and usually large)*

$$H = \begin{bmatrix} 1&1&1&1&1&1&0&0&0&0&\cdots&0&0&0&0&0&0 \\ 0&0&1&1&1&1&1&1&0&0&\cdots&0&0&0&0&0&0 \\ 0&0&0&0&1&1&1&1&1&1&\cdots&0&0&0&0&0&0 \\ \vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\ddots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots \\ 0&0&0&0&0&0&0&0&0&0&\cdots&1&1&1&1&1&1 \\ 1&1&0&0&0&0&0&0&0&0&\cdots&0&0&1&1&1&1 \\ 1&1&1&1&0&0&0&0&0&0&\cdots&0&0&0&0&1&1 \end{bmatrix} \in \mathbb{F}_2^{\frac{n}{2}\times n}. \tag{5.19}$$

This $[n, \frac{n}{2}]$-code is called a regular $(3, 6)$-LDPC code since there are always 3 (and 6) of '1's in each column (and rows) in its parity-check matrix.

From how the code is constructed, an LDPC code is nothing out of ordinary compared to a standard linear code. What really tells LDPC codes apart is their decoding method. In particular, the sparsity of H allows for a quite efficient iterative decoding heuristic called *belief propagation algorithm.* In the following sections, we introduce a graphical method known as *factor graphs* and the decoding algorithm for LDPC codes based on this graphical method.

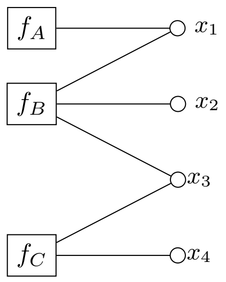
5.4.1 Factor graphs and graphical representation of linear codes

A factor graph is a graphical description of a factorization.

Example 5.13 (A simple factor graph). *Consider the factorization of function $g(x_1, x_2, x_3, x_4)$ as*

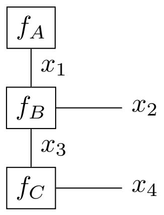
$$g(x_1, x_2, x_3, x_4) = f_A(x_1, x_3) \cdot f_B(x_1, x_2, x_3) \cdot f_C(x_3, x_4). \tag{5.20}$$

This factorization can be depicted by the following bipartite graph:



where we have used rectangular vertices (often referred to as "factor nodes") in representing each sub-functions and circular vertices (often referred to as "variable nodes") in representing each variables, and an edge connects a variable nodes to a factor nodes if and only if that variable is an argument of the sub-function. This graph is know as a factor graph (representing the aforementioned factorization).

If the degrees of all variable nodes are at most 2 (like in this case), it is often popular to simplify each variable node as edge or half edge:



This second graph is often known as a normal *factor graph.*

Formally, we define a factor graph as follows.

Definition 5.14 (Factor graph). *Given a factorization of function g as*

$$g(\{x_i\}_{i\in\mathcal{V}}) := \prod_{a\in\mathcal{F}} f_a(\{x_i\}_{i\in\partial a}) \tag{5.21}$$

where $\mathcal{V}$, $\mathcal{F}$ are some finite set, and $\partial a \subset V$ for each $a \in \mathcal{F}$. A factor graph *representing this factorization is the bipartite graph $(\mathcal{V}, \mathcal{F}, \mathcal{E})$ where the set of edges is defined as*

$$\mathcal{E} := \{(i,a) \in \mathcal{V}\times\mathcal{F} :\ i \in \partial a\}\,.$$

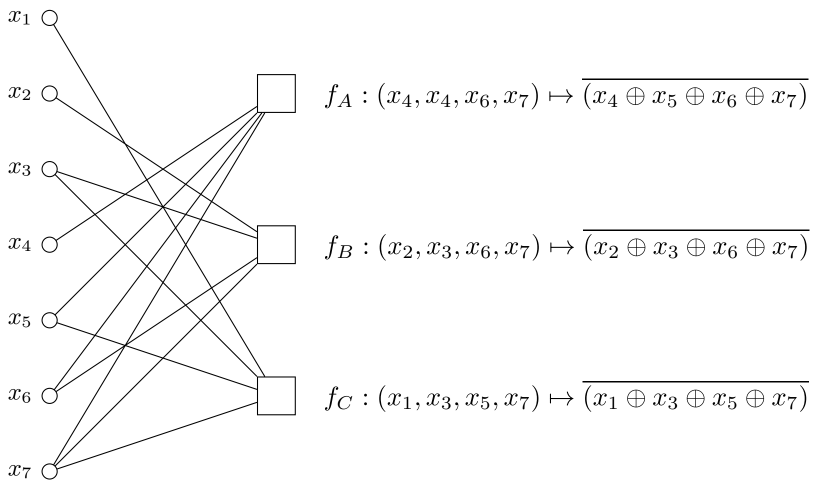
The function g is known as the global function *of the factor graph. If $\deg(i) \le 2$ for all $i \in \mathcal{V}$, such a factor graph is called a* normal *factor graph.*

Remark. Any factor graph can be converted into a normal factor graph by properly introduce some equality nodes.

Example 5.15 (Factor graph for the Hamming code). *Let us return to Example 5.9. By (5.9), A length-7 binary sequence $(x_1,\ldots,x_7)$ is a valid codeword if and only if $x_4 \oplus x_5 \oplus x_6 \oplus x_7 = 0$, $x_2 \oplus x_3 \oplus x_6 \oplus x_7 = 0$ and $x_1 \oplus x_3 \oplus x_5 \oplus x_7 = 0$. Thus, an* indicator *function telling whether $(x_1,\ldots,x_7)$ is a codeword can be expressed as follows*

$$\mathbf{1}_C(x_1,x_2,x_3,x_4,x_5,x_6,x_7) = \overline{(x_4 \oplus x_5 \oplus x_6 \oplus x_7)}\cdot\overline{(x_2 \oplus x_3 \oplus x_6 \oplus x_7)}\cdot\overline{(x_1 \oplus x_3 \oplus x_5 \oplus x_7)}$$

where $\overline{0} := 1$ and $\overline{1} := 0$. This expression can be visualized by the following factor graph.


x_1
x_2
x_3
x_4
x_5
x_6
x_7
$f_A : (x_4, x_4, x_6, x_7) \mapsto \overline{(x_4 \oplus x_5 \oplus x_6 \oplus x_7)}$
$f_B : (x_2, x_3, x_6, x_7) \mapsto \overline{(x_2 \oplus x_3 \oplus x_6 \oplus x_7)}$
$f_C : (x_1, x_3, x_5, x_7) \mapsto \overline{(x_1 \oplus x_3 \oplus x_5 \oplus x_7)}$

In general, given a parity-check matrix of a linear code, say $H \in \mathbb{F}^{(n-k)\times n}$. The indicator function for this code can be represented by a factor graph with n variable nodes and $n-k$ factor nodes. In particular, the degree of i-th variable/factor node equals to the number of nonzero entries in i-th column/row. For binary code, this graph fully defines the code (since it fully defines a parity-check matrix). In the remainder of this section, we will refer to this graph as a factor graph representing the code. A factor graph representing the LDPC code in Example 5.12 is depicted in Figure 5.4.1.

5.4.2 Use Belief-propagation algorithm to decode LDPC codes

In this section, we focus on the problem of decoding of LDPC codes sent through a memoryless channel. Given a code $\mathcal{C} \subset \mathbb{F}^n$, and a memoryless channel described by conditional pmf $p_{Y|X}$, supposing all codewords were sent with equal probability, and we received a n-length message $(y_1, \dots, y_n)$, the task of symbol-wise maximum-likelihood (ML) decoding is to find $x_i \in \mathcal{X}$ maximizing the marginals

$$\sum_{x_j : j \neq i} \prod_{i=1}^{n} p_{Y|X}(y_i|x_i) \cdot \mathbf{1}_{\mathcal{C}}(\boldsymbol{x}) \tag{5.22}$$

for each $i = 1, \dots, n$.

In general, computing the marginals of a function with n variables *directly* takes exponentially amount of time. On the other hand, if a factorization of the function is known, one could utilize distributivity, and, as illustrated in the following example, potentially compute faster.

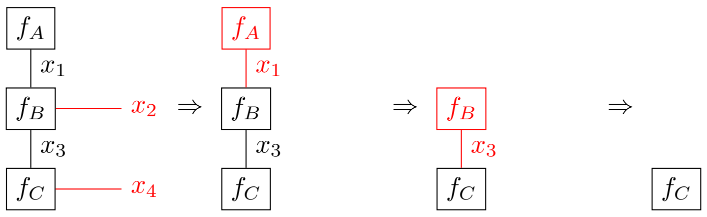
Example 5.16. *We are interested in computing*

$$Z = \sum_{x_1,x_2,x_3,x_4} f_A(x_1) \cdot f_B(x_1,x_2,x_3) \cdot f_C(x_3,x_4) \tag{5.23}$$

where $x_i \in \mathcal{X}$ for each i for some finite set $\mathcal{X}$. Direct computation takes $|\mathcal{X}|^4 - 1$ number of summations and $|\mathcal{X}|^4 + |\mathcal{X}|^3$ number of multipications. On the other hand, we could compute Z using $(|\mathcal{X}|+1)^2 \cdot (|\mathcal{X}|-1)$ number of summations and $|\mathcal{X}|^2 + |\mathcal{X}|$ number of multiplications as

$$Z = \sum_{x_3} \left[\sum_{x_1} f_A(x_1) \left(\sum_{x_2} f_B(x_1,x_2,x_3) \right) \right] \left(\sum_{x_4} f_C(x_3,x_4) \right). \tag{5.24}$$

Interestingly, this process can be visualized as a sequence of transformations on the factor graph representing (5.23):



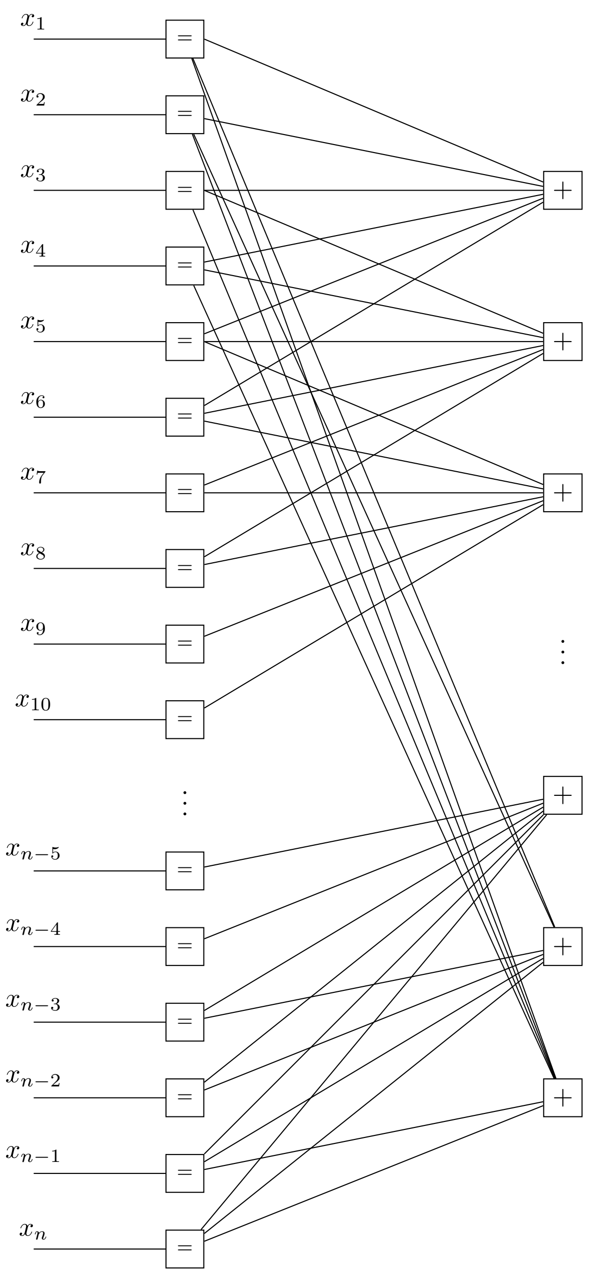


Figure 5.1: A normal factor graph representing the LDPC code in Example 5.12. Here, each '=' node corresponds to a local function that equals 1 only when all of the arguments are same (and equals 0 otherwise). Each + node corresponds to a local function that equals 1 only when the summation of all of the arguments is $0_{\mathbb{F}_2}$ (and equals 0 otherwise). This graph is also known as a Tanner graph.

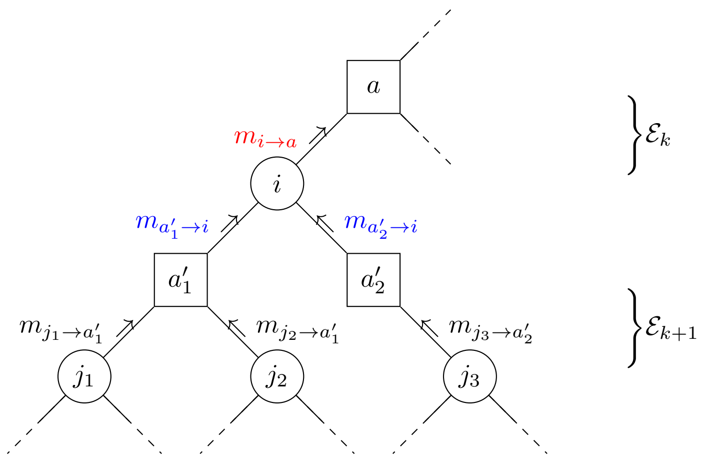


Figure 5.2: A step of updates of the messages as in Algorithm 5.1: Notice that the red message (output of this step) depends on the blue messages, and the blue messages depends on the black messages (outputs from the last step).

```
Data: Acyclic factor graph $\{\mathcal{V},\mathcal{F},\mathcal{E}\}$, alphabets $\mathcal{X}_i$ for each $i \in \mathcal{V}$, local funcitons
      $f_a : \times_{i:(i,a)\in\mathcal{V}}\mathcal{X}_i \to \mathbb{R}$ for each $a \in \mathcal{F}$.
Result: Partition $Z := \sum_{(x_i)_{i\in\mathcal{V}}} \prod_{a\in\mathcal{F}} f_a(\boldsymbol{x}_{\partial a})$ ;          ▶ $\partial a := \{i \in \mathcal{V} : (i,a) \in \mathcal{E}\}$
Pick $r \in \mathcal{F}$ as a root of the graph, and define set $\mathcal{E}_1 \leftarrow \{(i,r) : i \in \partial r\}$;
$k \leftarrow 1$, Height $\leftarrow 0$;          ▶ Partition the tree according to the depth
do
    Height $\leftarrow$ Height $+ 1$, $k \leftarrow k+1$, and $\mathcal{E}_k \leftarrow \emptyset$;
    foreach $(i,a) \in \mathcal{E}_{k-1}$ do
        foreach $a' \in \partial i \setminus \{a\}$ do
            $\mathcal{E}_k \leftarrow \mathcal{E}_k \cup \{(j,a') : j \in \partial a' \setminus \{i\}\}$;
        end
    end
while $\mathcal{E}_k \neq \emptyset$;
foreach $h =$ Height, Height $- 1, \ldots, 1$ do
    foreach $(i,a) \in \mathcal{E}_h$ do
        foreach $a' \in \partial i \setminus \{a\}$ do
            $m_{a'\to i}(x_i) \leftarrow \sum_{\boldsymbol{x}_{\partial a'\setminus\{i\}}} f_{a'}(\boldsymbol{x}_{\partial a'}) \cdot \prod_{j\in\partial a'\setminus\{i\}} m_{j\to a'}(x_j)$
        end
        $m_{i\to a}(x_i) \leftarrow \prod_{a'\in\partial i\setminus\{a\}} m_{a'\to i}(x_i)$;
    end
end
$Z \leftarrow \sum_{\boldsymbol{x}_{\partial r}} \prod_{i\in\partial r} m_{i\to r}(x_i)$;
```
Algorithm 5.1: Acyclic Belief-Propagation Algorithm

The above method can be systematically applied on any *acyclic* factor graphs (*i.e.* without cycles) by shrinking the graph down to its root as illustrated in the picture above. We formally describe this method as Algorithm 5.1. The key idea of this algorithm is to implement a sequence of instances of distributivity as a series of consecutive updates of messages alone the paths from the leaves of the graph to its root (see Figure 5.4.2), namely for each (i, a) along the paths:

$$m_{i\to a}(x_i) := \prod_{a'\in\partial i\setminus\{a\}} m_{a'\to i}(x_i), \tag{5.25}$$

$$m_{a\to i}(x_i) := \sum_{\boldsymbol{x}_{\partial\setminus\{i\}}} f_a(\boldsymbol{x}_{\partial a}) \cdot \prod_{j\in\partial a\in\setminus\{i\}} m_{j\to a}(x_j). \tag{5.26}$$

One must also note that we have applied "vacancy convention" for defining the initial messages from the leaves of the graph. Namely, for any $i \in \mathcal{V}$ with no children, $m_{i\to a}$ is defined as constant function 1 (a being its parent); whereas for any $i \in \mathcal{V}$ with no grandchildren, $m_{a'\to i} := f_{a'}$ (a' being its children). As a useful fact, this algorithm can also be used to compute the marginals by realizing that

$$\sum_{\boldsymbol{x}_{\mathcal{V}\setminus\partial r}} \prod_{a\in\mathcal{F}} f_a(\boldsymbol{x}_{\partial a}) \equiv \prod_{i\in\partial r} m_{i\to r}(x_i). \tag{5.27}$$

Unfortunately, factor graphs in most applications (including most LDPC codes) are *not* acyclic. Fortunately, however, for factor graphs with long *girth*[1] (including those of LDPC codes), the idea of updating the messages in Algorithm 5.1 (see (5.25), (5.26)) provides a nice heuristic in approximating the marginals of the global function. Namely, for each time stamp $t = 1, 2, \ldots$, we compute a new set of messages as

$$m^{(t)}_{i\to a}(x_i) \propto \prod_{a'\in\partial i\setminus\{a\}} m^{(t)}_{a'\to i}(x_i), \tag{5.28}$$

$$m^{(t)}_{a\to i}(x_i) \propto \sum_{\boldsymbol{x}_{\partial\setminus\{i\}}} f_a(\boldsymbol{x}_{\partial a}) \cdot \prod_{j\in\partial a\in\setminus\{i\}} m^{(t-1)}_{j\to a}(x_j), \tag{5.29}$$

and *hope* that the messages will converge as $t \to \infty$, and take

$$\prod_{a\in\partial i} m^{(\infty)}_{a\to i}(x_i) \Big/ \sum_{x} \prod_{a\in\partial i} m^{(\infty)}_{a\to i}(x) \tag{5.30}$$

as an estimated marginal of $\sum_{\boldsymbol{x}_{\mathcal{V}\setminus\partial r}} \prod_{a\in\mathcal{F}} f_a(\boldsymbol{x}_{\partial a}) / \sum_{\boldsymbol{x}} \prod_{a\in\mathcal{F}} f_a(\boldsymbol{x}_{\partial a})$. This method, known as the *belief propagation algorithm*, is formally described in Algorithm 5.2. Despite the heuristic nature of the belief propagation algorithm, the algorithm converges in many real life applications and often provides useful estimates to the marginals, especially when the factor graph in question is of long girth. The analysis of the performance of the belief

[1]The *girth* of a graph is the length of the shorted cycle in the graph.

```
Data: Factor graph $\{\mathcal{V}, \mathcal{F}, \mathcal{E}\}$, alphabets $\mathcal{X}_i$ for each $i \in \mathcal{V}$, local funcitons
      $f_a : \times_{i:(i,a)\in\mathcal{V}} \mathcal{X}_i \to \mathbb{R}$ for each $a \in \mathcal{F}$, a time limit $T \in \mathbb{N}$, and a tolerance
      coefficient $\epsilon \geq 0$.
Result: Estimates $\beta_i(x_i) \approx Z^{-1} \cdot \sum_{\boldsymbol{x}_{\mathcal{V}\setminus\{i\}}} \prod_{a\in\mathcal{F}} f_a(\boldsymbol{x}_{\partial a})$ for each $i \in \mathcal{V}$, and
        $\beta_a(\boldsymbol{x}_{\partial a}) \approx Z^{-1} \cdot \sum_{\boldsymbol{x}_{\mathcal{V}\setminus\partial a}} \prod_{a\in\mathcal{F}} f_a(\boldsymbol{x}_{\partial a})$ for each $a \in \mathcal{F}$, where
        $Z := \sum_{(x_i)_{i\in\mathcal{V}}} \prod_{a\in\mathcal{F}} f_a(\boldsymbol{x}_{\partial a})$.
foreach $(i,a) \in \mathcal{E}$ do
    $m^{(0)}_{i\to a}(x_i) \leftarrow |\mathcal{X}_i|^{-1}$ ;                    ▶ Initialization
end
$t \leftarrow 0$;
do
    $t \leftarrow t+1$;
    foreach $(i,a) \in \mathcal{E}$ do
        $m^{(t)}_{a\to i}(x_i) \leftarrow \sum_{\boldsymbol{x}_{\partial\setminus\{i\}}} f_a(\boldsymbol{x}_{\partial a}) \cdot \prod_{j\in\partial a\in\setminus\{i\}} m^{(t-1)}_{j\to a}(x_j)$ for each $x_i \in \mathcal{X}_i$;
        $m^{(t)}_{a\to i}(x_i) \leftarrow m^{(t)}_{a\to i}(x_i) \big/ \sum_{x\in\mathcal{X}_i} m^{(t)}_{a\to i}(x)$ for each $x_i \in \mathcal{X}_i$;
    end
    foreach $(i,a) \in \mathcal{E}$ do
        $m^{(t)}_{i\to a}(x_i) \leftarrow \prod_{a'\in\partial i\setminus\{a\}} m^{(t-1)}_{a'\to i}(x_i)$ for each $x_i \in \mathcal{X}_i$;
        $m^{(t)}_{i\to a}(x_i) \leftarrow m^{(t)}_{i\to a}(x_i) \big/ \sum_{x\in\mathcal{X}_i} m^{(t)}_{i\to a}(x)$ for each $x_i \in \mathcal{X}_i$;
    end
while $\exists (i,a) \in \mathcal{E}$ s.t. $\left\| m^{(t)}_{i\to a} - m^{(t-1)}_{i\to a} \right\|_1 > \epsilon$ or $\left\| m^{(t)}_{a\to i} - m^{(t-1)}_{a\to i} \right\|_1 > \epsilon$, and $t < T$;
foreach $i \in \mathcal{V}$ do
    $\beta_i(x_i) \leftarrow \prod_{a\in\partial i} m^{(t)}_{a\to i}(x_i)$ for each $x_i \in \mathcal{X}_i$;
    $\beta_i(x_i) \leftarrow \beta_i(x_i) \big/ \sum_{x\in\mathcal{X}_i} \beta_i(x)$ for each $x_i \in \mathcal{X}_i$;
end
foreach $a \in \mathcal{F}$ do
    $\beta_a(\boldsymbol{x}_{\partial a}) \leftarrow f_a(\boldsymbol{x}_{\partial a}) \cdot \prod_{i\in\partial a} m^{(t)}_{i\to a}(x_i)$ for each $\boldsymbol{x}_{\partial a} \in \times_{i\in\partial a}\mathcal{X}_i$;
    $\beta_a(\boldsymbol{x}_{\partial a}) \leftarrow \beta_a(\boldsymbol{x}_{\partial a}) \big/ \sum_{\boldsymbol{x}\in\times_{i\in\partial a}\mathcal{X}_i} \beta_a(\boldsymbol{x})$ for each $\boldsymbol{x}_{\partial a} \in \times_{i\in\partial a}\mathcal{X}_i$;
end
```

Algorithm 5.2: Belief-Propagation Algorithm

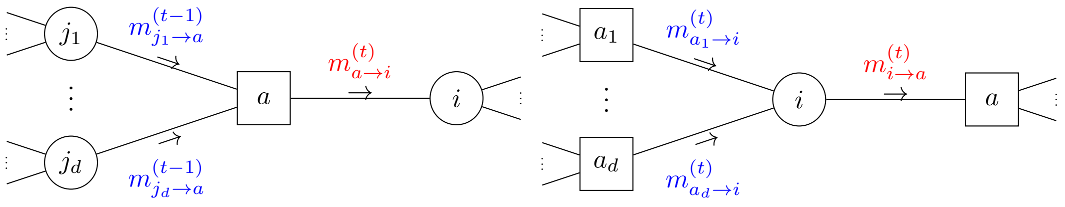


Figure 5.3: Message updating rules of the belief propagation algorithm: In both figures, the red message depends on the blue messages.

propagation algorithm is beyond the scope of this lecture. For interested readers, we refer to[**?**], and [**?**] for some of such studies.

We now move on to the decoding of LDPC codes using the belief propagation algorithm. Let $\mathcal{C}$ be an LDPC code of length n over $\mathbb{F}_2$ described by its parity-check matrix $H \in \mathbb{F}_2^{r\times n}$ $(1 < r < n)$. The indicator function for this code (see Example 5.15) can be expressed as

$$\mathbf{1}_{\mathcal{C}}(\boldsymbol{x}^n) = \prod_{a=1}^{r} \delta_{\sum_{i=1}^{n} H_{a,i}x_i,\,0} = \prod_{a=1}^{r} \left(\overline{\bigoplus_{i:H_{a,i}>0} x_i} \right). \tag{5.31}$$

Suppose we pick each of the codeword in $\mathcal{C}$ with equal probability and send the n symbols through n instances of a memoryless channel $W_{Y|X}$. The joint probability of inputs and outputs can be expressed as

$$P_{X^n,Y^n}(\boldsymbol{x}^n, \boldsymbol{y}^n) = \frac{1}{|\mathcal{C}|}\mathbf{1}_{\mathcal{C}}(\boldsymbol{x}^n) \cdot \prod_{i=1}^{n} W_{Y|X}(y_i|x_i). \tag{5.32}$$

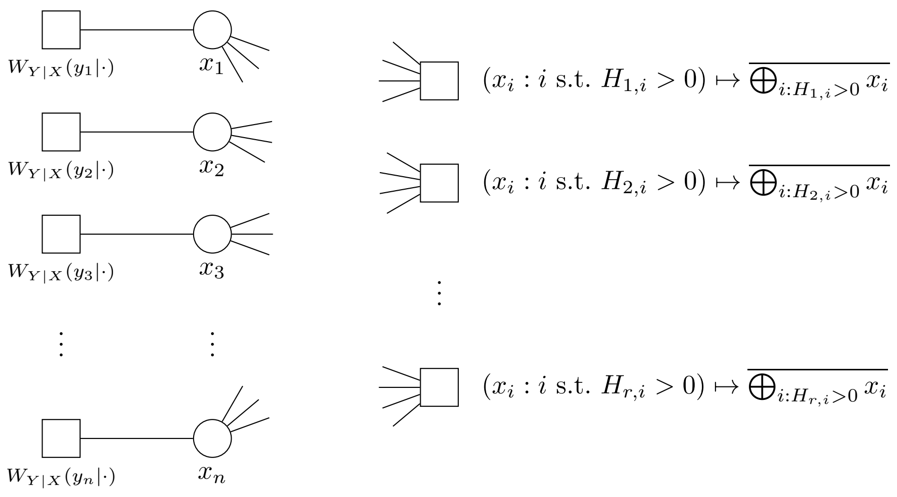
Since the conditional probability $P_{X^n|Y^n}(\cdot|\boldsymbol{y}^n)$ is proportional to $P_{X^n,Y^n}(\cdot, \boldsymbol{y}^n)$ for each fixed $\boldsymbol{y}^n$, the task of symbol-wise ML decoding is to pick each $\hat{x}_i \in \mathbb{F}_q$ as

$$\begin{aligned}\hat{x}_i \leftarrow \operatorname*{argmax}_{x_i} \sum_{\substack{x_1,\dots,x_{i-1},\\ x_{i+1},\dots,x_n}} P_{X^n|Y^n}(\boldsymbol{x}^n|\boldsymbol{y}^n) &= \operatorname*{argmax}_{x_i} \sum_{\substack{x_1,\dots,x_{i-1},\\ x_{i+1},\dots,x_n}} P_{X^n,Y^n}(\boldsymbol{x}^n, \boldsymbol{y}^n) \\ &= \operatorname*{argmax}_{x_i} \sum_{\substack{x_1,\dots,x_{i-1},\\ x_{i+1},\dots,x_n}} \prod_{a=1}^{r} \left(\overline{\bigoplus_{i:H_{a,i}>0} x_i} \right) \cdot \prod_{i=1}^{n} W_{Y|X}(y_i|x_i) \end{aligned} \tag{5.33}$$

for $i = 1, \dots, n$. This can be achieved by estimating the marginals in (5.33) rather efficiently using the belief propagation algorithm on the factor graph depicted in Figure 5.4.2. We formalize the discussion here in Algorithm 5.3. Note that in Algorithm 5.3, we did some simplification to avoid unnecessary updates of the messages.

5.4.3 Density evolution of LDPC codes for BECs

In this section, we analyze the performance of the decoding algorithm of regular binary LDPC codes when memoryless binary erasure channels (BECs) are used. In particular, we



$W_{Y|X}(y_1|\cdot)$
x_1
$W_{Y|X}(y_2|\cdot)$
x_2
$W_{Y|X}(y_3|\cdot)$
x_3
$W_{Y|X}(y_n|\cdot)$
x_n
$(x_i : i \text{ s.t. } H_{1,i} > 0) \mapsto \overline{\bigoplus_{i:H_{1,i}>0} x_i}$
$(x_i : i \text{ s.t. } H_{2,i} > 0) \mapsto \overline{\bigoplus_{i:H_{2,i}>0} x_i}$
$(x_i : i \text{ s.t. } H_{r,i} > 0) \mapsto \overline{\bigoplus_{i:H_{r,i}>0} x_i}$

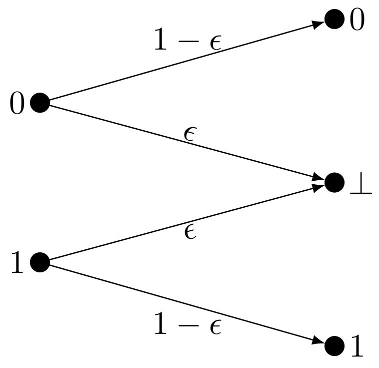
Figure 5.4: Factor graph used in the decoding of LDPC code

```
Data: An LDPC code over $\mathbb{F}_2$ described by its parity-check matrix $H \in \mathbb{F}_2^{r\times n}$,
      channel $W_{Y|X}$, channel output $\boldsymbol{y}^n$, a time limit $T \in \mathbb{N}$, and a tolerance
      coefficient $\epsilon \geq 0$.
Result: Estimated channel inputs $\hat{\boldsymbol{x}^n}$.
foreach $(i,a) \in [1:n] \times [1:r]$ s.t. $H_{a,i} = 1$ do
    $m^{(0)}_{i\to a}(x_i) \leftarrow 1/2$ for each $x_i \in \mathbb{F}_2$;
end
$t \leftarrow 0$;
do
    $t \leftarrow t+1$;
    foreach $(i,a) \in [1:n] \times \overline{[1:r]}$ s.t. $H_{a,i} = 1$ do
        $m^{(t)}_{a\to i}(x_i) \leftarrow \sum_{\boldsymbol{x}_{\partial a\setminus\{i\}}} \overline{\bigoplus_{i:H_{a,i}>0} x_i} \cdot \prod_{j\in\partial a\in\setminus\{i\}} m^{(t-1)}_{j\to a}(x_j)$ for each $x_i \in \mathbb{F}_2$;
        $m^{(t)}_{a\to i}(x_i) \leftarrow m^{(t)}_{a\to i}(x_i) / \sum_{x\in\mathcal{X}_i} m^{(t)}_{a\to i}(x)$ for each $x_i \in \mathbb{F}_2$;
    end
    foreach $(i,a) \in [1:n] \times [1:r]$ s.t. $H_{a,i} \neq 0$ do
        $m^{(t)}_{i\to a}(x_i) \leftarrow W_{Y|X}(y_i|x_i) \cdot \prod_{a'\in\partial i\setminus\{a\}} m^{(t-1)}_{a'\to i}(x_i)$ for each $x_i \in \mathbb{F}_2$;
        $m^{(t)}_{i\to a}(x_i) \leftarrow m^{(t)}_{i\to a}(x_i) / \sum_{x\in\mathcal{X}_i} m^{(t)}_{i\to a}(x)$ for each $x_i \in \mathbb{F}_2$;
    end
while $\exists(i,a)$ s.t. $\left\| m^{(t)}_{i\to a} - m^{(t-1)}_{i\to a} \right\|_1 > \epsilon$ or $\left\| m^{(t)}_{a\to i} - m^{(t-1)}_{a\to i} \right\|_1 > \epsilon$, and $t < T$;
foreach $i \in \{1,\dots,n\}$ do
    $\hat{x}_i \leftarrow \operatorname{argmax}_{x_i\in\mathbb{F}_q} \prod_{a\in\partial i} m^{(t)}_{a\to i}(x_i)$;
end
```

Algorithm 5.3: Decoding of an LDPC code over a memoryless channel: Notice that we define $\partial a := \{i \in \{1,\dots,n\} : H_{a,i} = 1\}$ and $\partial i := \{a \in \{1,\dots,r\} : H_{a,i} = 1\}$.

derive necessary condition for the decoding to succeed. Despite the restricted scenario, the demonstrated method has also been used in more generic cases.

A *binary erasure channel* (BEC) takes a binary input to a ternary output, $\{0,1,\perp\}$. The output $\perp$ has probability ϵ on either input, and otherwise the input symbol remains unaffected. Essentially this is a channel that flags errors:



The conditional probability distribution of the channel is given by

$$W_{\mathrm{BEC}(\epsilon)}(y|x) = (1-\epsilon)\delta_{x,y} + \epsilon\,\delta_{y,\perp}\,. \tag{5.34}$$

We make the following two important observations:

1. Suppose the received channel output is $\{y_1, y_2, \ldots, y_n\}$. At $t=1$, the messages $m^{(1)}_{i\to a}$ will become follows

$$m^{(1)}_{i\to a} = \begin{cases} (1,0) & \text{if } y_i = 0 \\ (0,1) & \text{if } y_i = 1 \\ (0.5,0.5) & \text{if } y_i = \perp \end{cases} \qquad \forall a \in \partial i$$

   for all $i = 1, \ldots, n$.

2. For any valid channel output $\{y_1, y_2, \ldots, y_n\}$, the messages $\{m^{(t)}_{i\to a}, m^{(t)}_{a\to i}\}_{i,a}$ are always among $\{(1,0),(0,1),(0.5,0.5)\}$ for all $t > 0$. In particular, we have following simplified message updating rules:

$$m^{(t)}_{a\to i} \leftarrow \begin{cases} (0.5,0.5) & \text{if } m^{(t-1)}_{j\to a} = (0.5,0.5) \text{ for some } j \in \partial a \setminus \{i\} \\ (1,0) & \text{if } \bigoplus_{j\in\partial a\setminus\{i\}} (m^{(t-1)}_{j\to a})^{-1}(1) = 0 \\ (0,1) & \text{if } \bigoplus_{j\in\partial a\setminus\{i\}} (m^{(t-1)}_{j\to a})^{-1}(1) = 1 \end{cases} \tag{5.35}$$

$$m^{(t)}_{i\to a} \leftarrow \begin{cases} (1,0) & \text{if } y_i = 0, \text{ or } m^{(t)}_{a'\to i} = (1,0) \text{ for some } a' \in \partial i \setminus \{a\} \\ (0,1) & \text{if } y_i = 1, \text{ or } m^{(t)}_{a'\to i} = (0,1) \text{ for some } a' \in \partial i \setminus \{a\} \\ (0.5,0.5) & \text{otherwise} \end{cases} \tag{5.36}$$

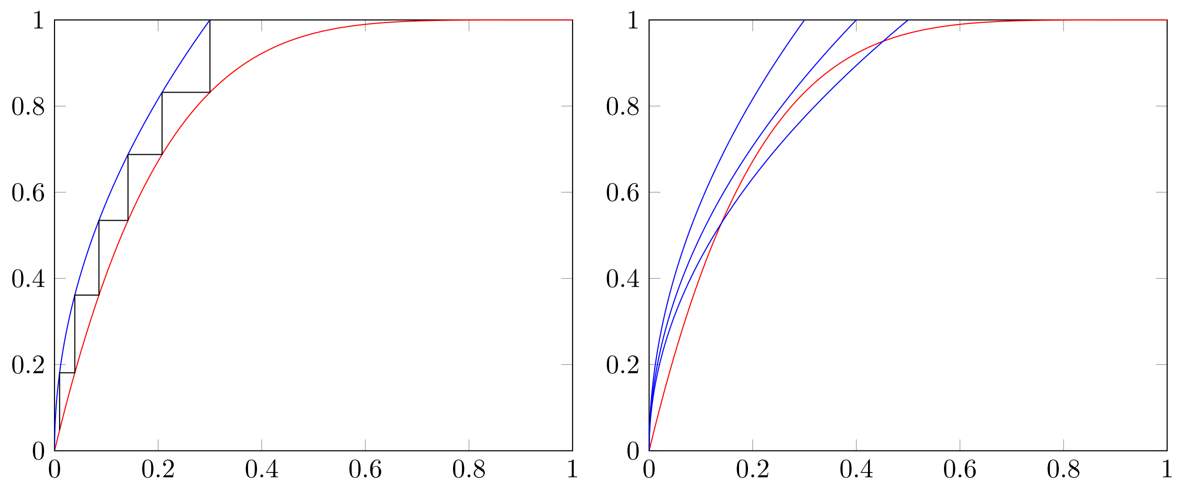


Figure 5.5: [Left-hand side] The development of $\{p_t\}_t$ and $\{q_t\}_t$ as t grows: The black curve corresponds to the curve $(p_1, q_1) - (p_1, q_2) - (p_2, q_2) - \cdots - (p_t, q_t) - \cdots$. In this particular plot, $w_R = 6$, $w_L = 3$, and $\epsilon = 0.3$. [Right-hand side] Plots of $y = f_1(x)$ and $x = f_2(y)$ for $\epsilon = 0.3$, 0.4, 0.5.

Consider a length-n regular (w_L, w_R)-LDPC code, *i.e.*, a binary code whose parity-check matrix has w_L '1's in each column and w_R '1's in each row. Suppose the n bits were sent through a BEC with erasure probability ϵ. Let p_t denote the probability that $m_{i \to a}^{(t)} = (0.5, 0.5)$. Let q_t denote the probability that $m_{a \to i}^{(t)} = (0.5, 0.5)$. The above two observations enable us to write

$$q_1 = 1,\ p_1 = \epsilon, \tag{5.37}$$

$$q_t = 1 - (1 - p_{t-1})^{w_R - 1} =: f_1(p_{t-1}), \tag{5.38}$$

$$p_t = \epsilon \cdot q_t^{w_L - 1} =: f_2(q_t). \tag{5.39}$$

By defining two functions $y = f_1(x) = 1-(1-x)^{w_R-1}$ and $x = f_2(y) = \epsilon \cdot y^{w_L-1}$, the sequences $\{p_t\}_t$ and $\{q_t\}_t$ can be visualized as 'bouncing' between the gap of the two function curves (*e.g.*, see LHS of Figure 5.4.3). On the RHS of the same figure, we plots several $x = f_2(y)$ for different ϵ. The plots *suggest* that the decoding will be successful if and only if the blue curve steers clear from the red curve. This matches with the intuition that the decoding is getting harder as the erasure ratio increases. More precisely, we have the following theorem.

Theorem 5.17 (Area theorem)**.** *Consider a length-n regular (w_L, w_R)-LDPC code sent through n-copies of BECs with erasure probability $\epsilon > 0$. Let $f_1(x) = 1 - (1 - x)^{w_R-1}$ and $f_2(y) = \epsilon \cdot y^{w_L-1}$. If the decoding using belief propagation is successful with probability* 1*, the area below $x = f_2(y)$ and above $y = f_1(x)$ must be non-negative.*

The proof of this theorem utilizes results from the next chapter.

Proof. By calculus, one can show that the area $= w_L^{-1}(C - R)$ where C is the capacity of the channel and R is the rate of the code. For decoding to be successful with probability 1, it must hold that $C \geq R$, and thus the area is non-negative. $\square$