

University of Oxford

Mathematical Institute

Information Theory

Ben Hambly [1]

Michaelmas Term 2025

[1]Notes adapted from those of Harald Oberhauser, Hanqing Jin and Sam Cohen.

Contents

Introduction **5**

1 Entropy, Divergence and Mutual Information **7**

1.1 Basic definitions in information theory 7

1.1.1 Conditional entropy/divergence/mutual information 11

1.2 Basic properties and inequalities 12

1.2.1 Divergence properties 13

1.2.2 Mutual information properties 14

1.2.3 Entropy properties 16

1.3 Fano's inequality 17

2 Codes and sequences **19**

2.1 The Kraft–McMillan Theorem 20

2.2 The Asymptotic Equipartition Property 21

2.3 Strong typicality 23

3 Optimal Codes and Shannon's first theorem **25**

3.1 Optimality of codes 25

3.2 Approaching the lower bound by block codes 27

3.3 Shannon's code (1948) 28

3.4 Fano's code (1949) (not examinable) 29

3.5 Huffman codes (1952) 30

3.6 Elias' code 33

3.7 Arithmetic codes (1970s) 33

3.8 Variable-to-fixed codes 34

3.8.1 Tunstall's code (1967) 35

4 Channel Coding: Shannon's Second Theorem **37**

4.1 Discrete memoryless channels . . . 37

4.2 Channel capacity . . . 38

4.3 Channel codes, rates and errors . . . 40

4.4 Shannon's second theorem: noisy channel coding . . . 41

4.5 Channel codes . . . 46

4.6 Block linear codes . . . 47

5 Noisy Channels with non-iid input **51**

5.1 Channel coding with non-iid input . . . 51

5.2 Combining symbol and channel coding for DMCs . . . 56

5.3 Decoding from a noisy non-iid channel . . . 58

5.3.1 Wonham filter . . . 58

5.3.2 Viterbi algorithm . . . 60

A Some useful probability theory **63**

A.1 Measure theory . . . 63

A.2 Probability spaces . . . 63

A.3 Discrete random variables . . . 64

A.4 Expectation . . . 65

A.5 Conditional probabilities and conditional expectations . . . 65

Introduction

Communication theory is a relatively young subject. It played an important role in the rise of the current information/digital/computer age and still motivates much research. Every time you make a phone call, store a file on your computer, query an internet search engine, watch a DVD, stream a movie, listen to a CD or mp3 file, etc., algorithms run that are based on topics we discuss in this course. However, independent of such applications, the underlying mathematical objects arise naturally as soon as one starts to think about "information", its representation and how to transfer and store information. In fact, a large part of the course deals with two fundamental questions:

(1) How much information is contained in a signal/data/message? (source coding)

(2) What are the limits to information transfer over a channel that is subject to noisy perturbations? (channel coding)

To answer these questions we will need to develop new mathematical concepts. These concepts also give new interpretations of important results in probability theory. Moreover, they are intimately connected to

- Physics: Thermodynamics, Statistical mechanics, Quantum theory,
- Computer Science: Kolmogorov complexity, etc.
- Statistics and Machine learning,
- Large deviation theory,
- Economics, finance, gambling.

Textbook and Literature. For most parts of the course we follow the classic textbook

- Cover, T. and Thomas, J. (2012). Elements of information theory. John Wiley & Sons.

Another excellent book is

- MacKay, D. J. (2003). Information theory, inference and learning algorithms. Cambridge University Press,

which has a more informal approach but many applications and is freely available on David MacKay's old webpage[2]. A concise treatment, focused on the theory is

- Csiszar, Körner (2011). Information Theory: Coding Theorems for Discrete Memoryless Systems. Cambridge University Press.

[2] https://www.inference.org.uk/mackay/itila/

Chapter 1

Entropy, Divergence and Mutual Information

Most of this course deals with sequences that are not perfectly predictable, for example, sequences of letters in texts or sequences of noisy observations of a system. In order to rigorously give a mathematical description of this randomness, we will use a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ (as in B8.1) where Ω is the set of all states of the world, $\mathcal{F}$ is the σ-algebra of events, and $\mathbb{P} : \mathcal{F} \to [0, 1]$ gives the probabilities of different events. On this space we define random variables, which are functions from $(\Omega, \mathcal{F})$ to $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$. We will usually not need this underlying structure and only talk about random variables. Furthermore, we will focus on discrete random variables which take values in a discrete subset $\mathcal{X} \subset \mathbb{R}$. This means that we need relatively little of the formal structures of measure theory (though some are discussed in the appendix), as we can describe the distributions of random variables by their probability mass functions (pmf), e.g., for a random variable X, its pmf is

$$p(x) = \mathbb{P}(X = x) \text{ for } x \in \mathcal{X}.$$

As X takes values in $\mathbb{R}$, we can also describe its distribution through its cumulative distribution function $x \mapsto \mathbb{P}(X \leq x)$. More generally, for multivariate random variables (taking values in $\mathbb{R}^n$) we can still describe their pmf (and various generalizations of cdf exist).

We will also write $P_X = \mathbb{P} \circ X^{-1}$ for the distribution (or 'law') of X, which is (formally) a map $(\mathbb{R}, \mathcal{B}(\mathbb{R})) \to [0, 1]$. Many properties of a random variable are determined by its law – for example, the expected value $\mathbb{E}[X]$ is determined by P_X, rather than being directly dependent on the random variable X. On the other hand, many interesting properties of random variables may be more subtle, for example independence of X and Y depends on the joint law of (X, Y), and not on the individual laws of X and Y.

1.1 Basic definitions in information theory

Our first challenge is to define the 'level of randomness' of a random variable. Here we use the notion of entropy which goes back in some form to Boltzmann, Gibbs and Maxwell's work on the statistical

mechanics of gasses (the name goes back in some form to Rudolph Clausius in 1865, and is a greek neologism (en-trope = εν-τροπη ≈ in-changing)). Claude Shannon adapted these tools to the understanding of random signals, which led to the development of information theory in the late 1940s. Von Neumann famously stated that "no one knows what entropy really is", and suggested Shannon use this name.

We seek to define $H(X)$, the entropy[1] of a random variable X. One way (among many!) to motivate the idea of entropy is to think of $H(X)$ as a measure of the average uncertainty we have about the value of X: the less certain we are, the bigger $H(X)$. To see this, we first derive a function $s(A)$ to measure the "surprise" of observing the event $\{X \in A\}$ for a set $A \subset \mathcal{X}$. It seems natural to demand that

(i) $s(A)$ depends continuously on $\mathbb{P}(X \in A)$,

(ii) $s(A)$ is decreasing in $\mathbb{P}(X \in A)$,

(iii) $s(A \cap B) = s(A) + s(B)$ whenever $\mathbb{P}(X \in A \cap B) = \mathbb{P}(X \in A)\mathbb{P}(X \in B)$, i.e., the surprise about the occurrence of two independent events $\{X \in A\}, \{X \in B\}$ is the sum of the surprises of each of these events.

Using that $\mathbb{P}(X \in A \cap B) = \mathbb{P}(X \in A)\mathbb{P}(X \in B)$, it follows that $s(A) = -\log(\mathbb{P}(A))$ fulfills these properties and is the unique function with these properties (up to choice of base of the logarithm, this is a result of Cauchy – see Cauchy's functional equation). In some books, $s(A)$ is also called the Shannon information content of the outcome A. This leads us to the following definition:

Definition 1.1 (Entropy)**.** *The entropy $H_b(X)$ in base b of a discrete random variable X is defined as*

$$H_b(X) = -\sum_{x \in \mathcal{X}} \mathbb{P}(X = x) \log_b \mathbb{P}(X = x), \tag{1.1.1}$$

where we use the convention that $0 \times \log_b(0) = 0$. As we will often work with binary signals, for $b = 2$ we usually write $H(X)$ instead of $H_2(X)$, and write $\log(q)$ instead $\log_2(q)$.

There are other motivations for the definition of $H(X)$ some of which we will see later (e.g. as a compression bound, as the average number of yes-no-questions to determine a value, etc).

Some remarks on this concept:

- The notation $H(X)$ is somewhat misleading, since the entropy only depends on the law of the random variable X, i.e., for two different random variables X and $\hat{X}$ with the same pmf, their entropies are the same. However, this notation is standard in the literature, and can be compared with $\mathbb{E}[X]$.

- Given the above point, for notational simplicity, it is also convenient to write $H(P_X)$ or $H(p_X)$ (i.e. just recording the law of X as an argument), and we observe this is the same as the entropy of X.

[1] Many have wondered why we use the letter H for entropy. Bolzmann originally used E for his related quantity, but it seems in the very early 1900s Gibbs and Zermelo (probably) chose to use a capital η (which looks like H) instead, for reasons which remain unknown... After suggestions by von Neumann, Shannon chose H to align with the standard usage by the 1940s, with reference to Bolzmann's H-theorem

- We can write[2] $H(X) = -\mathbb{E}[\log(p(X))]$ where $p(\cdot) = p_X(\cdot)$ is the pmf of X.
- The choice of base 2 for the logarithm is common (due to computers using two states) but not essential. Since $\log(x) = \log_b(x) = \frac{\log_a(x)}{\log_a(b)}$, we have $H_b(X) = \frac{1}{\log_a(b)} H_a(X)$.
- The unit of entropy in base 2 is called a `bit`, in base e `nat`, in base 256 a `byte`. Unless otherwise stated, we will take logarithms to base 2.

Example 1.2. If $\mathcal{X} = \{H, T\}$ and $\mathbb{P}(X = H) = p$, then

$$H(X) = -p\log(p) - (1-p)\log(1-p). \tag{1.1.2}$$

If $p \in \{0, 1\}$, then $H(X) = 0$. Differentiating in p shows that the entropy as a function of p increases on $(0, 0.5)$ and decreases on $(0.5, 1)$. Hence, the entropy is maximised if $p = 0.5$ with $H(X) = \log(2) = 1$ bits. In order to maximize notational confusion, as this case comes up frequently, we will also write

$$H(p) = -p\log(p) - (1-p)\log(1-p),$$

in which case we have the equivalent notations $H(X) = H(p) = H(p_X) = H(P_X)$.

Example 1.3. If X is a 2-dim vector in the form (X_1, X_2) with $X_i \in \mathcal{X}_i$ for $i = 1, 2$, then

$$H(X) = H(X_1, X_2) = -\sum_{x_1 \in \mathcal{X}_1, x_2 \in \mathcal{X}_2} p_{X_1,X_2}(x_1, x_2) \log\big(p_{X_1,X_2}(x_1, x_2)\big). \tag{1.1.3}$$

If additionally, X_1 and X_2 are independent, i.e., $p_{X_1,X_2}(x_1, x_2) = p_{X_1}(x_1)p_{X_2}(x_2)$, then

$$H(X) = H(X_1) + H(X_2). \tag{1.1.4}$$

If X_1 and X_2 are independent and identically distributed (i.i.d.), then

$$H(X) = 2H(X_1) = 2H(X_2). \tag{1.1.5}$$

Now assume, X models a coin flip as in Example 1.2, i.e. X takes values in $\mathcal{X} = \{H, T\}$. Given knowledge about p, we want to store the results of a sequence of n independent coin flips. One extreme case is $p \in \{0, 1\}$, in which case we need $H(X) = 0$ bits, the other extreme is $p = 0.5$ in which, by extending Example 1.3, it should be intuitive that we need n bits. This hints at another interpretation of entropy, namely as a storage/compression bound on information. We make this connection rigorous later in the course.

Definition 1.4 (Divergence)**.** *Let p and q be pmfs on $\mathcal{X}$. We call*

$$D(p\|q) = \sum_{x \in \mathcal{X}} p(x) \log\left(\frac{p(x)}{q(x)}\right) \tag{1.1.6}$$

the divergence between p and q and set by convention $0 \times \log(0) = 0$ and $D(p\|q) = \infty$ if $\exists x \in X$ such that $q(x) = 0$ and $p(x) > 0$. (Divergence is also known as information divergence, Kullback–Leibler divergence, relative entropy).

[2]Warning: one often uses x as an index for the pmf, i.e. $p_X(x) = \mathbb{P}(X = x)$. In this case the entropy is written as $H(X) = -\mathbb{E}[\log(p_X(X))] = -\sum_{x \in \mathcal{X}} p_X(x)\log(p_X(x))$ but we emphasise that $p_X : \mathcal{X} \mapsto [0, 1]$ is a function and not random (does not depend on $\omega \in \Omega$)! A better notation would be to enumerate r.v. values x_i with $i \in \mathbb{N}$ and to denote the pmf of X with $p_i = \mathbb{P}(X = x_i)$, though this is less standard.

Note that, given $X \sim p$ (which means the pmf of X is p),

$$\begin{aligned} D(p\|q) &= \mathbb{E}\left[\log\left(\frac{p(X)}{q(X)}\right)\right] \\ &= \mathbb{E}\left[\log\left(\frac{1}{q(X)}\right)\right] - \mathbb{E}\left[\log\left(\frac{1}{p(X)}\right)\right] \\ &= \mathbb{E}\left[\log\left(\frac{1}{q(X)}\right)\right] - H(X). \end{aligned}$$

Note that while we will show below that divergence is always non-negative it is not a metric: in general it is not symmetric and can take the value ∞. These properties are actually useful and desirable as the following example shows.

Example 1.5. (Asymmetry and infinite values are useful). Let $\mathcal{X} = \{0, 1\}$ and $p(0) = 0.5$, $q(0) = 1$. We are given independent samples from one of these two distributions but we do not know which one. If we observe 0000001, we can immediately infer that p is the underlying pmf. On the other hand, if we observe 0000000 it is likely that the sample comes from q but we cannot exclude that it comes from p. This is reflected in the divergence since $D(p\|q) = \infty$ but $D(q\|p) = 1$.

Example 1.6. If q is a uniform distribution, then it is clear that the relative entropy

$$D(p\|q) = \log(|\mathcal{X}|) - H(X).$$

Definition 1.7 (Mutual Information)**.** *Let X, Y be discrete random variables taking values in $\mathcal{X}$ and $\mathcal{Y}$ respectively. The mutual information $I(X;Y)$ between X and Y is defined as*

$$I(X;Y) = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} \mathbb{P}(X = x, Y = y) \log\left(\frac{\mathbb{P}(X = x, Y = y)}{\mathbb{P}(X = x)\mathbb{P}(Y = y)}\right).$$

Some motivations:

- Denote by $p_{X,Y}, p_X, p_Y$ the pmfs of $(X, Y), X$ and Y. Then

  $$I(X;Y) = D(p_{X,Y}\|p_X p_Y).$$

  Hence, we can regard the mutual information as a measure of how much dependence there is between two random variables.

- Unlike covariance $\mathrm{Cov}(X, Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$, the mutual information $I(X;Y)$ takes into account higher order dependence (not just second order dependence).

- It is obvious that $I(X;Y) = I(Y;X)$.

- Another way to think about mutual information is in terms of entropies

  $$\begin{aligned} I(X;Y) &= \mathbb{E}\left[\log\left(\frac{p_{X,Y}(X,Y)}{p_X(X)p_Y(Y)}\right)\right] = \mathbb{E}[\log(p_{X,Y}(X,Y)) - \log(p_X(X)) - \log(p_Y(Y))] \\ &= H(X) + H(Y) - H(X,Y). \end{aligned}$$

1.1.1 Conditional entropy/divergence/mutual information

Often we are given additional knowledge, for instance the outcome of another random variable. This motivates us to generalise the concepts of entropy, divergence and information to the situation where we condition on this extra information.

Definition 1.8 (Conditional entropy)**.** *Let X, Y be discrete random variables taking values in $\mathcal{X}$. The conditional entropy of Y given X is defined as*

$$\begin{aligned} H(Y|X) &= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{X}} \mathbb{P}(X=x, Y=y)\log\big(\mathbb{P}(Y=y|X=x)\big) \\ &= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{X}} \mathbb{P}(X=x, Y=y)\log\Big(\frac{\mathbb{P}(Y=y, X=x)}{\mathbb{P}(X=x)}\Big). \end{aligned}$$

In analogy to entropy, we have

$$\begin{aligned} H(Y|X) &= -\sum_{x\in\mathcal{X}} \mathbb{P}(X=x)\sum_{y\in\mathcal{X}} \mathbb{P}(Y=y|X=x)\log\big(\mathbb{P}(Y=y|X=x)\big) \\ &= \sum_{x\in\mathcal{X}} \mathbb{P}(X=x)H(Y|X=x) \\ &= -\sum_{x\in\mathcal{X}} \mathbb{P}(X=x)\mathbb{E}[\log(p_{Y|X=x}(Y))] \\ &= -\mathbb{E}[\log(p_{Y|X}(Y))]. \end{aligned}$$

An intuitive way to think about $H(Y|X)$ is as the average (with respect to X) surprise we have about Y after having observed X (e.g. if $Y = X$ there is no surprise).

By rearranging and Bayes' rule, we have the 'chain rule' of conditional entropy

$$H(Y|X) = H(X,Y) - H(X).$$

Definition 1.9 (Conditional divergence)**.** *Let p_X be a pmf on a discrete space $\mathcal{X}$, and $p(\cdot|x)$ and $q(\cdot|x)$ be two (conditional on the parameter x) pmfs on $\mathcal{X}$ for any $x \in \mathcal{X}$. The divergence between $p(\cdot|X)$ and $q(\cdot|X)$ conditioned on p_X (also known as conditional divergence, conditional Kullback–Leibler divergence, conditional relative entropy) is defined as*

$$D(p_{Y|X}\|q_{Y|X}\,|\,p_X) = \sum_{x\in\mathcal{X}} p_X(x)D(p_{Y_1|X=x}\|q_{Y_2|X=x})$$

where random variables X, Y, Y_1, Y_2 are all such that $p_{Y|X}(y|x) = p(y|x) = p_{Y_1|X}(y|x)$, $q_{Y|X}(y|x) = q(y|x) = q_{Y_2|X}(y|x)$.

We could also give a version of the definition in terms of random variables.

Definition 1.10 (Conditional divergence (v2))**.** *Let X and Y_1, Y_2 be discrete random variables taking values in $\mathcal{X}$, and the joint pmf of (X, Y_i) be p_{X,Y_i}. The divergence between p_{Y_1} and p_{Y_2} conditioned on X is defined as*

$$D(p_{Y_1|X}\|p_{Y_2|X}\,|\,p_X) = \sum_{x\in\mathcal{X}} p_X(x)D(p_{Y_1|X=x}\|p_{Y_2|X=x})$$

and can be written

$$D(p_{Y|X}\|q_{Y|X}\,|\,p_X) = \mathbb{E}[D(p_{Y_1|X}(\cdot|X)\|p_{Y_2|X}(\cdot|X))].$$

Definition 1.11 (Conditional mutual information)**.** *Let X, Y, Z be discrete random variables taking values in $\mathcal{X}$. The conditional mutual information $I(X;Y|Z)$ (conditioned on Z) between X and Y is defined as*

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

Again, we can write this as $I(X;Y|Z) = \mathbb{E}\left[\log\left(\frac{p_{X,Y|Z}(X,Y)}{p_{X|Z}(X)p_{Y|Z}(Y)}\right)\right]$, by which we can see that $I(X;Y|Z) = I(Y;X|Z)$.

In the same way we regard mutual information as measure of dependence, we can regard conditional mutual information as a measure of dependence of two r.v.'s (X,Y) conditional on knowing another random variable (Z).

1.2 Basic properties and inequalities

We prove some basic properties of entropy, divergence and mutual information. We prepare this with three elementary but important inequalities.

Firstly we introduce convex functions.

Definition 1.12. *Let $I \subset \mathbb{R}$. We call $f : I \longrightarrow \mathbb{R}$ convex, if*

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$$

for all $x, y \in I$ and $\lambda \in [0,1]$. We call f strictly convex if this is a strict inequality for all $\lambda \in (0,1)$.

A convex function f lies below any of its chords. A function f is concave if $-f$ is convex. We state a fundamental inequality for convex functions in our discrete space setting (see B8.1 for the general case).

Lemma 1.13. *(Jensen's inequality) If f is a convex function and X a random variable taking values in $\mathcal{X}$ with pmf p, such that $\mathbb{E}X = \sum_{x\in\mathcal{X}} xp(x)$ and $\mathbb{E}f(X) = \sum_{x\in\mathcal{X}} f(x)p(x)$ exist, then*

$$f\left(\sum_{x\in\mathcal{X}} xp(x)\right) = f(\mathbb{E}(X)) \leq \mathbb{E}(f(X)) = \sum_{x\in\mathcal{X}} f(x)p(x).$$

If f is strictly convex, then equality holds if and only if X is constant.

Proof. This is proved firstly for a two point pmf where it is just the definition of convexity. It is then a straightforward induction to prove it for any discrete state space - see Problem Sheet 1. □

Lemma 1.14. *(Gibbs' inequality) Let p and q be pmfs on $\mathcal{X}$. Then*

$$-\sum_{x\in\mathcal{X}} p(x)\log(p(x)) \leq -\sum_{x\in\mathcal{X}} p(x)\log(q(x))$$

and the equality holds if and only if (iff) $p \equiv q$.

Proof. Denote by X a random variable with the pmf p. Adding $\sum_{x\in\mathcal{X}} p(x)\log(p(x))$ to both sides, we estimate the right hand side

$$\begin{aligned}\sum_{x\in\mathcal{X}} p(x)\log\left(\frac{p(x)}{q(x)}\right) &= \mathbb{E}\left[-\log\left(\frac{q(X)}{p(X)}\right)\right]\\ &\geq -\log\left(\mathbb{E}\left[\frac{q(X)}{p(X)}\right]\right)\\ &= -\log\left(\sum_{x\in\mathcal{X}} p(x)\frac{q(x)}{p(x)}\right)\\ &= -\log(1) = 0.\end{aligned}$$

where the inequality follows by Jensen's inequality applied to $f(x) = -\log(x)$ (a strictly convex function). Note that, by Jensen, equality holds iff $\frac{q(x)}{p(x)}$ is constant. □

Put differently, Gibbs' inequality tells us that the minimiser of the map

$$q \mapsto -\mathbb{E}[\log(q(X))]$$

is the pmf p_X and the minimum is $H(X)$.

Lemma 1.15. *(Log sum inequality) Let $a_1,\cdots,a_n; b_1,\cdots,b_n$ be nonnegative. Then*

$$\sum_{i=1}^{n} a_i\log\left(\frac{a_i}{b_i}\right) \geq \left(\sum_{i=1}^{n} a_i\right)\log\left(\frac{\sum_{i=1}^n a_i}{\sum_{i=1}^n b_i}\right)$$

with equality holds iff $\frac{a_i}{b_i}$ is constant.

1.2.1 Divergence properties

Theorem 1.16. *(Divergence properties). Let (X,Y) and $(\tilde{X},\tilde{Y})$ be 2-dimensional discrete random variables taking values in $\mathcal{X}\times\mathcal{Y}$. Then*

(1) (Information inequality) $D(p_X\|p_{\tilde{X}}) \geq 0$ with equality iff $p_X = p_{\tilde{X}}$.

(2) (Chain rule) $D(p_{X,Y}\|p_{\tilde{X},\tilde{Y}}) = D(p_{Y|X}\|\, p_{\tilde{Y}|\tilde{X}}\,|\,p_X) + D(p_X\|p_{\tilde{X}})$.

(3) $D(p_{X,Y}\|p_{\tilde{X},\tilde{Y}}) \geq D(p_X\|p_{\tilde{X}})$.

(4) $D(p_{Y|X}\|p_{\tilde{Y}|\tilde{X}}\,|\,p_X) = D(p_Xp_{Y|X}\|p_Xp_{\tilde{Y}|\tilde{X}})$.

(5) (Convexity) For pmfs p_1,p_2,q_1,q_2, we have $D(\lambda p_1+(1-\lambda)p_2\|\lambda q_1+(1-\lambda)q_2) \leq \lambda D(p_1\|q_1)+(1-\lambda)D(p_2\|q_2)$ for $\forall\,\lambda\in[0,1]$.

Proof. (1) follows from Gibbs' inequality; (2) follows from

$$\begin{aligned}
D(p_{X,Y}\|p_{\tilde{X},\tilde{Y}}) &= \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p_{X,Y}(x,y)\log\left(\frac{p_{X,Y}(x,y)}{p_{\tilde{X},\tilde{Y}}(x,y)}\right)\\
&= \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p_{X,Y}(x,y)\log\left(\frac{p_X(x)p_{Y|X}(y|x)}{p_{\tilde{X}}(x)p_{\tilde{Y}|\tilde{X}}(y|x)}\right)\\
&= \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p_{X,Y}(x,y)\log\left(\frac{p_{Y|X}(y|x)}{p_{\tilde{Y}|\tilde{X}}(y|x)}\right) + \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p_{X,Y}(x,y)\log\left(\frac{p_X(x)}{p_{\tilde{X}}(x)}\right)\\
&= \sum_{x\in\mathcal{X}} p_X(x)\sum_{y\in\mathcal{Y}} p_{Y|X}(y|x)\log\left(\frac{p_{Y|X}(y|x)}{p_{\tilde{Y}|\tilde{X}}(y|x)}\right) + D(p_X\|p_{\tilde{X}})\\
&= \sum_{x\in\mathcal{X}} p_X(x)D(p_{Y|X=x}\|p_{\tilde{Y}|\tilde{X}=x}) + D(p_X\|p_{\tilde{X}})\\
&= D(p_{Y|X}\|p_{\tilde{Y}|\tilde{X}})\,|p_X) + D(p_X\|p_{\tilde{X}}).
\end{aligned}$$

With (2), and the fact $D(p_1\|p_2|p) \geq 0$ for any pmf's p_1, p_2, q, we have (3).

(4) follows since

$$\begin{aligned}
D(p_{Y|X}\|p_{\tilde{Y}|\tilde{X}}\,|\,p_X) &= \sum_{x\in\mathcal{X}} p_X(x)\sum_{y\in\mathcal{Y}} p_{Y|X}(y|x)\log\left(\frac{p_{Y|X}(y|x)}{p_{\tilde{Y}|\tilde{X}}(y|x)}\right)\\
&= \mathbb{E}\left[\log\left(\frac{p_{Y|X}(Y|X)}{p_{\tilde{Y}|\tilde{X}}(Y|X)}\right)\right]\\
&= \mathbb{E}\left[\log\left(\frac{p_X(X)p_{Y|X}(Y|X)}{p_X(X)p_{\tilde{Y}|\tilde{X}}(Y|X)}\right)\right]\\
&= D(p_X p_{Y|X}\,\|\,p_X p_{\tilde{Y}|\tilde{X}}).
\end{aligned}$$

For (5), we just need to apply Lemma 1.15 with $a_1 = \lambda p_1(x), a_2 = (1-\lambda)p_2(x), b_1 = \lambda q_1(x), b_2 = (1-\lambda)q_2(x)$ and then sum over $x \in \mathcal{X}$.

□

1.2.2 Mutual information properties

We give some key results about mutual information, including a chain rule and the data processing inequality. In order to state this we recall that X and Z are conditionally independent given Y (denoted by $(X\perp Z)\,|\,Y$) if $p_{(X,Z)|Y}(x,z|y) = p_{X|Y}(x|y)p_{Z|Y}(z|y)$. This is equivalent to stating that (X,Y,Z) is a Markov process with 3 time points, which can be described by $p_{X,Y,Z}(x,y,z) = p(x)p(y|x)p(z|y)$.

Theorem 1.17. *(Mutual Information properties).*

(1) $I(X;Y) \geq 0$ *with equality iff* $X\perp Y$ *(i.e.* X *and* Y *are independent)*

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

(3) (Information chain rule)

$$I(X_1,\cdots,X_n;Y) = \sum_{i=1}^{n} I(X_i;Y\,|\,X_{i-1},\cdots,X_1).$$

(4) (Data-processing inequality) If $(X \perp Z) \,|\, Y$*, then*

$$I(X;Y) \geq I(X;Z).$$

(5) Let $f : \mathcal{Y} \mapsto \mathcal{Z}$*. Then* $I(X;Y) \geq I(X;f(Y))$*.*

Proof. (1) follows from $I(X;Y) = D(p_{X,Y} \| p_X p_Y) \geq 0$ by the information inequality in Theorem 1.16. The first equality in (2) follows from the definition of mutual information. The others follow since

$$\begin{aligned} I(X;Y) &= \mathbb{E}\left[\log\left(\frac{p_{X,Y}(X,Y)}{p_X(X)p_Y(Y)}\right)\right] \\ &= H(X) + H(Y) - H(X,Y), \end{aligned}$$

and

$$\begin{aligned} H(X,Y) &= -\sum_{x\in\mathcal{X}, y\in\mathcal{Y}} \mathbb{P}(X=x, Y=y)\log(\mathbb{P}(Y=y, X=x)) \\ &= -\sum_{x\in\mathcal{X}, y\in\mathcal{Y}} \mathbb{P}(X=x, Y=y)[\log(\mathbb{P}(Y=y|X=x)) + \log(\mathbb{P}(X=x))] \\ &= H(Y|X) + H(X). \end{aligned}$$

Notice that the last equality can be easily extended to

$$H(X_1, \cdots, X_n) = H(X_n|X_{n-1}, \cdots, X_1) + H(X_{n-1}, \cdots, X_1) = \sum_{i=1}^{n} H(X_i|X_{i-1}, \cdots, X_1).$$

with the notation $H(X_1|X_0) = H(X_1)$. Furthermore, we can have the conditional version

$$H(X_1, \cdots, X_n \,|\, Y) = \sum_{i=1}^{n} H(X_i|X_{i-1}, \cdots, X_1, Y).$$

(3) follows from

$$\begin{aligned} I(X_1, \cdots, X_n; Y) &= H(X_1, \cdots, X_n) - H(X_1, \cdots X_n \,|\, Y) \\ &= \sum_{i=1}^{n} \{H(X_i|X_{i-1}, \cdots, X_1) - H(X_i|X_{i-1}, \cdots, X_1, Y)\} \\ &= \sum_{i=1}^{n} I(X_i; Y \,|\, X_{i-1} \cdots, X_1), \end{aligned}$$

where the last line uses the definition of conditional entropy. For (4) we use the chain rule (3) to write $I(Y,Z;X) = I(Y;X) + I(Z;X|Y) = I(Y;X)$. On the other hand, $I(Y,Z;X) = I(Z,Y;X) = I(Z;X) + I(Y;X|Z) \geq I(Z;X)$, so $I(X;Y) \geq I(X;Z)$.

Finally, (5) follows from the data-processing inequality in Point (4) by taking $Z = f(Y)$. □

Remark 1.18.

- Point (2) applied with $X = Y$ shows $I(X;X) = H(X)$ which explains why entropy is sometimes referred to as self-information.

- (2) also motivates us to think of $I(X;Y)$ as a measure of the reduction in uncertainty that knowing either variable gives about the other.
- Despite its simple form and proof, the data processing inequality in (4) and (5) formalises the intuitive but fundamental concept: post-processing cannot increase information; e.g., if Z is a r.v. that depends only on Y, then Z cannot contain more information about X than Y does.
- Recall from Statistics that an estimator $T(X)$ for a parameter $\theta \in \Theta$ is called sufficient if, conditional on $T(X)$, the distribution of X does not depend on θ. This is equivalent to $I(\theta; X) = I(\theta; T(X))$ under all distributions in $\{p_\theta : \theta \in \Theta\}$.

1.2.3 Entropy properties

Theorem 1.19. *(Entropy properties). Let X, Y be discrete random variables taking values in $\mathcal{X}$. Write $|\mathcal{X}|$ for the number of elements in $\mathcal{X}$.*

(1) $0 \leq H(X) \leq \log(|\mathcal{X}|)$. *The upper bound is attained iff X is uniformly distributed on X, the lower bound is attained iff X is constant with probability 1.*

(2) $0 \leq H(X|Y) \leq H(X)$ *and* $H(X|Y) = H(X)$ *iff X and Y are independent,* $H(X|Y) = 0$ *iff.* $X = f(Y)$ *for some function f.*

(3) (Chain rule) $H(X_1, \cdots, X_n) = \sum_{i=1}^n H(X_i|X_{i-1}, \cdots, X_1) \leq \sum_{i=1}^n H(X_i)$ *with equality iff the X_i are independent.*

(4) For $f : \mathcal{X} \mapsto \mathcal{Y}$, $H(f(X)) \leq H(X)$ *with equality iff f is injective (or one-to-one).*

(5) Let X and Y be i.i.d., then

$$\mathbb{P}(X = Y) \geq 2^{-H(X)}$$

with equality iff they are uniformly distributed.

(6) $H(X)$ *is concave in* p_X.

Proof. For (1), the lower bound follows from the definition of entropy; for the upper bound, we apply Gibbs' inequality with $q(x) = |\mathcal{X}|^{-1}$ to get

$$H(X) \leq -\sum_{x \in \mathcal{X}} p(x) \log(q(x)) = \log(|\mathcal{X}|).$$

Since equality holds in Gibbs' inequality iff $p_X = q$, it follows that X must be uniformly distributed to attain the upper bound. Similarly, since each term in the sum is non-positive (as $\log p \leq 0$) and is zero iff $p(x) = 0$ or $p(x) = 1$ and there can be just one x with $p(x) = 1$, we see that X must be constant to have zero entropy.

For (2), we use that $0 \leq I(X;Y) = H(X) - H(X|Y)$ by Theorem 1.17 so both bounds follow. The upper bound is attained iff X, Y are independent. For the lower bound, note that by definition

$$H(X|Y) = \sum_{y \in \mathcal{Y}} p_Y(y) H(X|Y = y),$$

where $H(X|Y=y) = -\sum_{x\in\mathcal{X}} p_{X|Y}(x|y)\log(p_{X|Y}(x|y))$. Hence, $H(X|Y)=0$ iff $H(X|Y=y)=0$ for all y in the support of Y. But by Point(1) this only happens if $\mathbb{P}(X=x|Y=y)=1$ for some constant $x=f(y)$. This implies that $X=f(Y)$.

(3) follows as in the proof of the Point (3) in Theorem 1.17, and the fact that $H(X_i|X_{i-1},\cdots,X_1) = H(X_i)$ iff X_i and $X_{i-1},\cdots,X_1$ are independent.

(4) follows since

$$H(X,f(X)) = H(X) + H(f(X)|X) = H(X)$$

and

$$H(X,f(X)) = H(f(X),X) = H(f(X)) + H(X|f(X)) \geq H(f(X)).$$

So $H(f(X)) \leq H(X)$, and the equality holds iff $H(X|f(X))=0$, which is equivalent to stating that f is injective.

(5) follows from Jensen's inequality,

$$2^{-H(X)} = 2^{\mathbb{E}[\log(p_X(X))]} \leq \mathbb{E}[2^{\log(p_X(X))}] = \mathbb{E}[p_X(X)] = \sum_{x\in\mathcal{X}} p_X(x)p_X(x) = \mathbb{P}(X=Y).$$

(6) follows from $g(x) = -x\log x$ is a concave function over $x\in(0,1)$. □

Remark 1.20.

- Point (1) is especially intuitive if we think of entropy as the average surprise we have about X.
- Point (2) formalises "more information is better".
- Point (4) shows that entropy is invariant under relabelling of observations.

1.3 Fano's inequality

A common situation is that we use an observation of a random variable Y to infer the value of a random variable X. If $\mathbb{P}(X\neq Y)=0$, then $H(X|Y)=0$ by Point (2) in Theorem 1.19. We expect that if $\mathbb{P}(X\neq Y)$ is small, then $H(X|Y)$ should be small. Fano's inequality makes this precise.

Theorem 1.21. *(Fano's inequality, 1966). Let X,Y be discrete random variables taking values in $\mathcal{X}$. Then*

$$H(X|Y) \leq H(\mathbf{1}_{X\neq Y}) + \mathbb{P}(X\neq Y)\log(|\mathcal{X}|-1).$$

Alternatively we can interpret Fano's inequality as giving a lower bound on the error probability $\mathbb{P}(X|Y)$ and this is how we will get bounds on the transmission of information over noisy channels.

Proof. Set $Z=\mathbf{1}_{X\neq Y}$ and note that $H(Z|X,Y)=0$. Now

$$\begin{aligned}
H(X|Y) &= H(X|Y) + H(Z|X,Y) \\
&= H(X,Z|Y) \\
&= H(Z|Y) + H(X|Y,Z) \\
&\leq H(Z) + H(X|Y,Z) \\
&= H(Z) + \sum_{y\in\mathcal{X}} [\mathbb{P}(Y=y,Z=0)H(X|Y=y,Z=0) + P(Y=y,Z=1)H(X|Y=y,Z=1)].
\end{aligned}$$

Now $\{Y = y, Z = 0\}$ implies $\{X = y\}$, hence $H(X|Y = y, Z = 0) = 0$. On the other hand, $\{Y = y, Z = 1\}$ implies that $\{X \in \mathcal{X}\backslash\{y\}\}$ which contains $|\mathcal{X}| - 1$ elements. Therefore,

$$H(X|Y = y, Z = 1) \leq \log(|\mathcal{X}| - 1).$$

It follows that

$$\begin{aligned} H(X|Y) &\leq H(Z) + \sum_{y \in \mathcal{X}} \mathbb{P}(Y = y, Z = 1) H(X|Y = y, Z = 1) \\ &\leq H(Z) + \mathbb{P}(Z = 1) \log(|\mathcal{X}| - 1). \end{aligned}$$

□

Corollary 1.22. $H(X|Y) \leq 1 + \mathbb{P}(X \neq Y) \log(|\mathcal{X}| - 1)$.

Chapter 2

Codes and sequences

Suppose that we receive a sequence from a set $\mathcal{X}$ (e.g. a sequence of letters from the latin alphabet) and we want to store this message, e.g. on our computer, so using a sequence of 0s and 1s.

Definition 2.1. *For a finite set $\mathcal{X}$, denote by $\mathcal{X}^*$ the set of finite sequences (also called strings) in $\mathcal{X}$. For $x = x_1 \cdots x_n \in \mathcal{X}^*$ with $x_i \in \mathcal{X}$ for all $i = 1, \cdots, n$, we call $|x| = n$ the length of the sequence $x \in \mathcal{X}^*$. Given two finite sets $\mathcal{X}$ and $\mathcal{Y}$, we call a function $c : \mathcal{X} \longrightarrow \mathcal{Y}^*$ a symbol code, and call $c(x) \in \mathcal{Y}^*$ the codeword of $x \in \mathcal{X}$. In this context, $\mathcal{Y}$ is called d-ary if $|\mathcal{Y}| = d$.*

That is, to encode $\mathcal{X}$, we look for a map $c : \mathcal{X} \longrightarrow \mathcal{Y}^*$ that allows us to recover any sequence in $\mathcal{X}^*$ from the associated sequence in $\mathcal{Y}^*$. We do this by concatenating the encodings of each source term, giving the sequence $c(x_1) \cdots c(x_n) \in \mathcal{Y}^*$. Since we need to recover the original sequence $x_1 \cdots x_n \in \mathcal{X}^*$, we clearly need to restrict attention to codes c that are injective. However, this is not sufficient.

Example 2.2. Let $\mathcal{X} = \{1, \cdots, 6\}$ and $c(x)$ be the binary expansion, i.e. the source code is a binary code with codewords $\{1, 10, 11, 100, 101, 110\}$. In general, we can not recover the original sequence, e.g. 110 might correspond to $x_1 = 6$ or $x_1 x_2 = 12$.

Ideally, we are looking for a code that allows us to recover the original message, and it is easy to decode in practice and compresses the original message as much as possible. A second objective is to make the encoding rules simple to construct and store. To make all this rigorous, we define different classes of codes.

Definition 2.3. *Let $c : \mathcal{X} \longrightarrow \mathcal{Y}^*$ be a symbol code. We denote by $c^* : \mathcal{X}^* \longrightarrow \mathcal{Y}^*$ the extension of c to $\mathcal{X}^*$ given by $c^*(x_1 \cdots x_n) = c(x_1) \cdots c(x_n)$. We say that c is*

(1) unambiguous (or nonsingular) if c is injective, so every $x \in \mathcal{X}$ maps to a different element of $\mathcal{Y}^*$,

(2) uniquely decodable if c^* is injective, so every sequence of characters in $\mathcal{X}$ maps to a different element of $\mathcal{Y}^*$ (without needing to separate characters!)

(3) a prefix code (or instantaneous code), if no codeword of c is the prefix of another codeword of c. That is, there does not exist $x_1 \in \mathcal{X}, x_2 \in \mathcal{X}$ such that $c(x_1)y = c(x_2)$ for some $y \in \mathcal{Y}^*$.

Clearly,

$$\{\text{prefix codes}\} \subset \{\text{uniquely decodable codes}\} \subset \{\text{unambigiuous codes}\}.$$

In general it is not easy to check if a given code is uniquely decodable; moreover, even if a code is uniquely decodable it can be very difficult/computationally expensive to decode.

Example 2.4. Take $\mathcal{X} = \{A, B, C, D\}, \mathcal{Y} = \{0, 1\}$. Then $c(A) = 0, c(B) = 01, c(C) = 011, c(D) = 111$ is uniquely decodable although this is not completely trivial to see. Note that describing a decoding algorithm is not easy either. For example, what leads to the string 011 111 01? What about 011 111 11?

2.1 The Kraft–McMillan Theorem

Prefix codes are easy to decode. A surprising result is that we can restrict attention to the design of prefix codes without increasing the length of code words.

Theorem 2.5 (Kraft–McMillan Theorem)**.** *(1) Let $c : \mathcal{X} \longrightarrow \mathcal{Y}^*$ be uniquely decodable and set $l_x = |c(x)|$. Then*

$$\sum_{x\in\mathcal{X}} |\mathcal{Y}|^{-l_x} \leq 1. \tag{2.1.1}$$

(2) Conversely, given $(l_x)_{x\in\mathcal{X}} \subset \mathbb{N}$ and a finite set $\mathcal{Y}$ such that (2.1.1) holds, there exists a prefix code $c : \mathcal{X} \longrightarrow \mathcal{Y}^*$ such that $|c(x)| = l_x$ for each $x \in \mathcal{X}$.

Proof. Set $d = |\mathcal{Y}|$ and $l_{\max} = \max_{x\in\mathcal{X}} |c(x)|$. We consider the source strings of length n, and obtain

$$\begin{aligned}\left(\sum_{x\in\mathcal{X}} d^{-l_x}\right)^n &= \sum_{x_1,x_2,\ldots,x_n\in\mathcal{X}} d^{-l_{x_1}} d^{-l_{x_1}} \cdots d^{-l_{x_n}} \\ &= \sum_{x_1,x_2,\ldots,x_n\in\mathcal{X}} d^{(-\sum_{i=1}^n l_{x_i})}.\end{aligned}$$

If we collect together output strings of length $k = \sum_{i=1}^n l_{x_i}$ for each k, and write $a(k)$ for the number of source sequences (of any length) mapping to codewords of length k, then we have

$$\left(\sum_{x\in\mathcal{X}} d^{-l_x}\right)^n \leq \sum_{k=1}^{nl_{max}} a(k)d^{-k}.$$

As there are d^k strings in $\mathcal{Y}^*$ of length k, unique decodability and the pigeonhole principle implies $a(k) \leq d^k$, hence $\sum_{x\in\mathcal{X}} d^{-l_x} \leq (nl_{max})^{1/n}$ for every n. Letting $n \to +\infty$ gives the first result.

To show the converse, let $(l_x)_{x\in\mathcal{X}}$ be a set of integers that fulfils (2.1.1). By relabelling, identify $\mathcal{X}$ as the set $\{1, \cdots, |\mathcal{X}|\} \subset \mathbb{N}$ and assume $l_1 \leq l_2 \leq \cdots \leq l_{|\mathcal{X}|}$. Define $r_m := \sum_{i=1}^{m-1} d^{-l_i}$ for any $m \leq |\mathcal{X}|$, which satisfies $r_m \leq 1$ by assumption. Define $c(m)$ to be the first l_m digits in the d-ary expansion[1] of the real number $r_m \in [0, 1)$, that is $\mathrm{c}(m) := y_1 \cdots y_{l_m}$, where

$$r_m = \sum_{i\geq 1} y_i d^{-i}.$$

[1]With the usual convention that an infinite number of zeros appears if needed to avoid ambiguity, e.g. with $d = 2$, we say that $\frac{1}{2}$ has the expansion $2^{-1} = 0.1000...$ and not $\sum_{i=2}^{+\infty} 2^{-i} = 0.0111....$

This must be a prefix code: if not, there exists m, n with $m < n$, and $c(m)$ a prefix of $c(n)$ and therefore the first l_m digits of r_m and r_n in the d-ary expansion coincide, which in turn implies $r_n - r_m < d^{-l_m}$; on the other hand, by the very definition of r_m and r_n we have $r_n - r_m = \sum_{i=m}^{n-1} d^{-l_i} \geq d^{-l_m}$, which is a contradiction. □

Remark 2.6. The inequality (2.1.1) is called the Kraft–McMillan inequality. Under the stronger assumption that p is a prefix code in Point (1), the above Theorem 2.5 has a nice proof using trees; Kraft showed the above theorem under this extra assumption. Theorem 2.5 as stated above is due to McMillan (based on Kraft's work).

Remark 2.7. We will see the construction used in Point (2) many times in the future!

Corollary 2.8. *For any uniquely decodable code there exists a prefix code with the same codeword lengths.*

2.2 The Asymptotic Equipartition Property

Given a discrete distribution, what can we infer about one sample from this distribution? Not much! An elementary but far reaching insight of Shannon is that this changes drastically if we deal with sequences of observations because of the law of large numbers. As we will be encoding sequences, this is quite useful!

Example 2.9. Denote by X a discrete random variable with state space $\mathcal{X} = \{0, 1\}$ and $X_1, \cdots, X_n$, i.i.d. copies of X. A sequence $(x_1, \cdots, x_n) \in \{0, 1\}^n$ occurs with probability

$$\mathbb{P}((X_1, \cdots, X_n) = (x_1, \cdots, x_n)) = p^{\alpha(x_1, \cdots, x_n)} q^{\beta(x_1, \cdots, x_n)}, \tag{2.2.1}$$

where $p = \mathbb{P}(X = 0)$, $q = 1 - p$ and $\alpha(x_1, \cdots, x_n) = \sum_i \mathbf{1}_{x_i=0}$, $\beta(x_1, \cdots, x_n) = \sum_i \mathbf{1}_{x_i=1}$. By the law of large numbers, for a "typical sequence" $(x_1, \cdots, x_n)$, we can approximate the numbers of 0's and 1's using $\alpha(x_1, \cdots, x_n) \approx \mathbb{E}[\alpha(X_1, \cdots, X_n))] = np$ and $\beta(x_1, \cdots, x_n) \approx \mathbb{E}[\beta(X_1, \cdots, X_n))] = nq$. Hence,

$$\mathbb{P}((X_1, \cdots, X_n) = (x_1, \cdots, x_n)) \approx p^{np} q^{nq}.$$

Taking the logarithm on both sides of these approximation, we get

$$-\log(\mathbb{P}((X_1, \cdots, X_n) = (x_1, \cdots, x_n))) \approx -np \log(p) - nq \log(q) = nH(X).$$

Thus for a "typical sequence" $(x_1, \cdots, x_n) \in \{0, 1\}^n$,

$$\mathbb{P}((X_1, \cdots, X_n) = (x_1, \cdots, x_n)) \sim 2^{-nH(X)}.$$

Therefore the set of typical sequences of length n consists of approximately $2^{nH(X)}$ elements, each occurring with approximate probability $2^{-nH(X)}$. Finally, note that $2^{nH(X)} \leq 2^n$, and this difference can be very large.

The above informal calculation suggests that we partition $\mathcal{X}^n$ into two sets,

- "typical sequences" and
- "atypical sequences".

The set of "typical sequences" forms a potentially relatively small subset of $\mathcal{X}^n$, that however carries most of the probability mass and its elements occur with approximately the same probability. In the rest of this section, we extend and make the above informal discussion rigorous.

Theorem 2.10 (Weak AEP 1)**.** *Let X be a discrete random variable. If $(X_i)_{i\geq 1}$ is an i.i.d sequence of samples of X, then*

$$-\frac{1}{n}\log(p_{X_1,\cdots X_n}(X_1,\cdots X_n)) \overset{\text{in prob.}}{\longrightarrow} H(X) \qquad \text{as } n\to+\infty. \tag{2.2.2}$$

Proof. By independence, $-\log(p_{X_1,\cdots X_n}(X_1,\cdots X_n)) = -\sum_{i=1}^n \log(p_X(X_i))$ and $\mathbb{E}[-\log(p_X(X_i))] = H(X)$. The result follows from the (weak) law of large numbers. □

Theorem 2.10 suggests the following definition of "typical sequences".

Definition 2.11. *For any $n\in\mathbb{N}$, any $\varepsilon>0$, we call*

$$\mathcal{T}_n^\varepsilon := \left\{(x_1,\cdots,x_n)\in\mathcal{X}^n : \left|-\frac{1}{n}\log(p_{X_1,\cdots X_n}(x_1,\cdots x_n)) - H(X)\right| \leq \varepsilon\right\}$$

the set of (weakly) typical sequences of length n of the random variable X (with error ε).

Theorem 2.12 (Weak AEP 2 (Shannon 1948))**.** *For all $\varepsilon>0$, there exists an $n_0\in\mathbb{N}$ such that for every $n>n_0$,*

(1) $p_{X_1,\cdots X_n}(x_1,\cdots x_n)\in[2^{-n(H(X)+\varepsilon)}, 2^{-n(H(X)-\varepsilon)}]$ for any $(x_1,\cdots,x_n)\in\mathcal{T}_n^\varepsilon$;

(2) $\mathbb{P}((X_1,\cdots,X_n)\in\mathcal{T}_n^\varepsilon)\geq 1-\varepsilon$;

(3) $|\mathcal{T}_n^\varepsilon|\in[(1-\varepsilon)2^{n(H(X)-\varepsilon)}, 2^{n(H(X)+\varepsilon)}]$.

Moreover, for Point (1) one can take $n_0=0$.

Proof. Point (1) follows directly from Definition 2.11 for $n_0=0$. Point (2) follows by Theorem 2.10, since for every $\varepsilon>0$,

$$\mathbb{P}((X_1,\cdots,X_n)\notin\mathcal{T}_n^\varepsilon) = \mathbb{P}\left(|\log p_{X_1,\cdots,X_n}(X_1,\cdots,X_n) - H(X)|>\varepsilon\right),$$

which converges to 0 as $n\to+\infty$.

For the upper bound in Point (3), observe that

$$\begin{aligned} 1 &= \sum_{(x_1,\cdots,x_n)\in\mathcal{X}^n} p_{X_1,\cdots,X_n}(x_1,\cdots,x_n) \\ &\geq \sum_{(x_1,\cdots,x_n)\in\mathcal{T}_n^\varepsilon} p_{X_1,\cdots,X_n}(x_1,\cdots,x_n) \\ &\geq \sum_{(x_1,\cdots,x_n)\in\mathcal{T}_n^\varepsilon} 2^{-n(H(X)+\varepsilon)} = 2^{-n(H(X)+\varepsilon)}|\mathcal{T}_n^\varepsilon|. \end{aligned}$$

For the lower bound, we know by Point (2) that the probability $\mathbb{P}((X_1,\cdots,X_n)\in\mathcal{T}_n^\varepsilon)$ converges to 1, so for n large enough,

$$1-\varepsilon\leq\mathbb{P}((X_1,\cdots,X_n)\in\mathcal{T}_n^\varepsilon)\leq\sum_{(x_1,\cdots,x_n)\in\mathcal{T}_n^\varepsilon} 2^{-n(H(X)-\varepsilon)} = 2^{-n(H(X)-\varepsilon)}|\mathcal{T}_n^\varepsilon|,$$

and then we get the lower bound. □

Remark 2.13. The AEP can be interpreted in a variety of ways:

- When n is large, the above result suggests that we think of $(X_1, \cdots, X_n)$ as being drawn uniformly from $\mathcal{T}_n^\varepsilon$ with probability $2^{-nH(X)}$.
- Theorem 2.12 does not imply that most sequences are elements of $\mathcal{T}_n^\varepsilon$, in fact $\mathcal{T}_n^\varepsilon$ has rather small cardinality compared to $\mathcal{X}^n$ since

$$\frac{|\mathcal{T}_n^\varepsilon|}{|\mathcal{X}^n|} \approx \frac{2^{nH(X)}}{2^{n\log(|\mathcal{X}|)}} = 2^{-n(\log(|\mathcal{X}|)-H(X))},$$

  and the last ratio converges to 0 when $n \to +\infty$ unless $H(X) = \log(|\mathcal{X}|)$, which holds iff X is uniformly distributed by Theorem 1.19. However, $\mathcal{T}_n^\varepsilon$ carries most of the probability mass, as shown in Point (2) in Theorem 2.12.
- Theorem 2.12 allows us to prove a property for typical sequences and then conclude that this property holds for random sequences $(X_1, \cdots, X_n)$ with high probability.
- The most likely sequence $x^* = \text{argmax}_x \mathbb{P}((X_1, \cdots, X_n) = x)$ is in general not an element of $\mathcal{T}_n^\varepsilon$. For example, take $\mathcal{X} = \{0, 1\}$ and $\mathbb{P}(X = 1) = 0.9$, then $(1, \cdots, 1)$ is the most likely sequence but not typical (for small ε) since $-\frac{1}{n}\log\left(p_{X_1,\cdots,X_n}(1, ..., 1)\right) = -\log(0.9) \approx 0.11$ is not close to $H(X) = -0.1\log(0.1) - 0.9\log(0.9) \approx 0.46$. Note that as $n \to +\infty$, the probability of every sequence, thus also the most likely sequence, tends to 0.

2.3 Strong typicality

The final point above suggests that we could ask if there are sets with smaller cardinality than $\mathcal{T}_n^\varepsilon$ that still carry most of the probability mass.

Definition 2.14. *Denote by $\mathcal{S}_n^\varepsilon$ the smallest subset of $\mathcal{X}^n$ such that*

$$\mathbb{P}((X_1, \cdots, X_n) \in \mathcal{S}_n^\varepsilon) \geq 1 - \varepsilon.$$

We can construct this set by ordering sequences by their probability and adding them until the probability mass is greater than or equal to $1 - \varepsilon$.

Proposition 2.15. *Let $(\varepsilon_m)_m$ be a strictly positive sequence such that $\lim_{m\to+\infty} \varepsilon_m = 0$. Then*

$$\lim_{m\to\infty} \left\{ \lim_{n\to\infty} \frac{1}{n} \log\left(\frac{|\mathcal{S}_n^{\varepsilon_m}|}{|\mathcal{T}_n^{\varepsilon_m}|} \right) \right\} = 0.$$

Proof (Sketch). We write ϵ for ϵ_m. We observe that $|\mathcal{T}_n^\varepsilon|$ is larger than $|\mathcal{S}_n^\varepsilon|$, with small error. Now show that $\mathcal{S}_n^\varepsilon$ and $\mathcal{T}_n^\varepsilon$ overlap apart from a set with probability $\leq 2\varepsilon$. The elements of $\mathcal{T}_n^\varepsilon$ have probabilities bounded above by $2^{-n(H-\varepsilon)}$, so the probability of $\mathcal{S}_n^\varepsilon \cap \mathcal{T}_n^\varepsilon$ is bounded above by $|\mathcal{S}_n^\varepsilon \cap \mathcal{T}_n^\varepsilon| 2^{-n(H-\varepsilon)}$. Therefore, by the weak AEP 2,

$$1 - 2\varepsilon \leq |\mathcal{S}_n^\varepsilon| 2^{-n(H-\varepsilon)} \leq \frac{|\mathcal{S}_n^\varepsilon|}{|\mathcal{T}_n^\varepsilon|} 2^{2n\epsilon}.$$

Rearranging and taking the limit, first in n and then down the sequence $\varepsilon = \varepsilon_m$, gives the result. □

In other words, the set of strong and weak typical sequences have the same number of elements up to first order in the exponent. Hence, we do not gain by working with strong typical sequences instead of weak typical sequences although the construction of $\mathcal{S}_n^\varepsilon$ appears at first sight to be more efficient than that of $\mathcal{T}_n^\varepsilon$. Nevertheless, one could argue that the definition of $\mathcal{S}_n^\varepsilon$ is simpler and that we should prefer to work with $\mathcal{S}_n^\varepsilon$ instead of $\mathcal{T}_n^\varepsilon$. However, note that the proof relies on counting the elements of the set of "typical sequences": using $\mathcal{T}_n^\varepsilon$ this is trivial due to the "uniform distribution" of elements in $\mathcal{T}_n^\varepsilon$, but this is much harder for $\mathcal{S}_n^\varepsilon$.

Chapter 3

Optimal Codes and Shannon's first theorem

3.1 Optimality of codes

In this chapter we want to use symbol codes to compress, that is to associate with every element of $\mathcal{X}$ a sequence of bits (or more generally, a sequence of elements in a given set).

Definition 3.1. *We call a symbol code $c : \mathcal{X} \longrightarrow \mathcal{Y}^*$ optimal for a random variable X with pmf p on $\mathcal{X}$ and a finite set $\mathcal{Y}$, if it minimises $\mathbb{E}[|c'(X)|]$ among all uniquely decodable codes $c' : \mathcal{X} \longrightarrow \mathcal{Y}^*$.*

In view of the Kraft–McMillan inequality, given a set $\mathcal{Y}$, a code $c : \mathcal{X} \longrightarrow \mathcal{Y}^*$ is optimal if it solves the constrained minimisation problem

$$\begin{array}{ll} \text{Minimise} & \sum_{x\in\mathcal{X}} p(x) l_x \\ \text{s.t.} & \sum_{x:p(x)>0} d^{-l_x} \leq 1 \text{ and } (l_x)_{x\in\mathcal{X}} \subset \mathbb{N}. \end{array} \tag{3.1.1}$$

This is an integer programming problem, and such problems are in general (computationally) hard to solve. To get an idea about what to expect, let us first neglect the integer constraint $l_x \in \mathbb{N}$ and assume $\sum d^{-l_x} = 1$. This gives a simple optimisation problem that can be solved using Lagrange multipliers, i.e. differentiating

$$\sum_{x\in\mathcal{X}} p(x) l_x - \lambda \left(\sum_{x\in\mathcal{X}} d^{-l_x} - 1 \right)$$

with respect to l_x and setting the derivative to 0 and using the constraint gives $l_x = -\log_d(p(x))$ and it remains to verify that this is indeed a minimum. Still ignoring the integer constraint, this yields an expected length $\mathbb{E}[|c(X)|] = -\sum p(x) \log_d(p(x)) = H_d(X)$. Instead of using Lagrange multipliers we make this rigorous using a direct argument involving just basic properties of entropy and divergence from Chapter 1.

Theorem 3.2. *(Source coding for symbol codes). Let X be a random variable taking values in a finite set $\mathcal{X}$ and c a uniquely decodable, d-ary source code. Then*

$$H_d(X) \leq \mathbb{E}[|c(X)|],$$

and the equality holds iff $|c(x)| = -\log_d(\mathbb{P}(X = x))$.

Proof. Recall $l_x := |c(x)|$ and set $q(x) = \frac{d^{-l_x}}{\sum_{x\in\mathcal{X}} d^{-l_x}}$, then

$$\begin{aligned}
\mathbb{E}[|c(X)|] - H_d(X) &= \sum_{x\in\mathcal{X}} p(x) l_x + \sum_{x\in\mathcal{X}} p(x)\log_d(p(x)) \\
&= -\sum_{x\in\mathcal{X}} p(x)\log_d(d^{-l_x}) + \sum_{x\in\mathcal{X}} p(x)\log_d(p(x)) \\
&= -\sum_{x\in\mathcal{X}} p(x)\log_d\left(q(x)\sum_{x'\in\mathcal{X}} d^{-l_{x'}}\right) + \sum_{x\in\mathcal{X}} p(x)\log_d(p(x)) \\
&= -\sum_{x\in\mathcal{X}} p(x)\log_d\left(\sum_{x'\in\mathcal{X}} d^{-l_{x'}}\right) + \sum_{x\in\mathcal{X}} p(x)\log_d\left(\frac{p(x)}{q(x)}\right) \\
&= -\log_d\left(\sum_{x'\in\mathcal{X}} d^{-l_{x'}}\right) + D_d(p\|\, q) \\
&\geq 0,
\end{aligned}$$

where we have used that the divergence is non-negative and, by the Kraft–McMillan inequality (2.1.1), we have that $\sum_{x'\in\mathcal{X}} d^{-l_{x'}} \leq 1$. Equality holds iff $\sum_{x'\in\mathcal{X}} d^{-l_{x'}} = 1$ and $D(p\|\, q) = 0$. Since $D(p\|\, q) = 0$ implies $p = q$, the result follows by definition of q. □

Proposition 3.3. *Let X be a random variable taking values in a finite set $\mathcal{X}$, and $\mathcal{Y}$ a d-ary set. There exists an optimal code $\bar{c}$, which is prefix, for which*

$$H_d(X) \leq \mathbb{E}[|\bar{c}(X)|] < H_d(X) + 1. \tag{3.1.2}$$

Proof. Set $l_x := \lceil -\log_d(p(x))\rceil$, where $\lceil x\rceil$ is the ceiling function (the lowest integer not less than x), and note that $\sum_{x\in\mathcal{X}} d^{-l_x} \leq \sum_{x\in\mathcal{X}} d^{-(-\log_d(p(x)))} = \sum_{x\in\mathcal{X}} p(x) = 1$. By Kraft–McMillan (Theorem 2.5), there exists a (not necessarily optimal) prefix code c with word lengths $(l_x)_{x\in\mathcal{X}}$. Now by definition

$$-\log_d(p(x)) \leq l_x < -\log_d(p(x)) + 1,$$

and we can conclude by multiplying this inequality with $p(x)$ and summing over $x \in \mathcal{X}$ to get (3.1.2). This shows there is a prefix code with expected length less than $H_d(X) + 1$.

As $\mathcal{X}$ is finite, we know $\bar{p} := \min_{x:p(x)>0} p(x) > 0$, and hence, for every $x \in \mathcal{X}$ with $p(x) > 0$,

$$p(x) l_x \leq \mathbb{E}[|\bar{c}(X)|] < H_d(X) + 1.$$

Thus all codes with expected length less than $H_d(X) + 1$ must encode every symbol $x \in \mathcal{X}$ with $p(x) > 0$ using less than $(H_d(X) + 1)/\bar{p}$ symbols. There are finitely many codes of this type (if we ignore the encoding of states with $p(x) = 0$), and we have seen there exists such a code, so we can sort them by expected length and take a code that achieves the minimum expected length. By Kraft–McMillan we can assume this code is prefix. The lower bound is given by Theorem 3.2. □

3.2 Approaching the lower bound by block codes

If c is an optimal code we are only guaranteed that

$$H_d(X) \leq \mathbb{E}[|c(X)|] < H_d(X) + 1.$$

The overhead of 1 is negligible if X has high entropy but it can be the dominating term for low entropies. We also only have an existence result, with no way (yet) of constructing good codes.

A code in which we encode fixed length blocks is called a block code. By using such a code we can get arbitrary close to the lower bound: if we apply Proposition 3.3 to the $\mathcal{X}^n$-valued random variable $(X_1, \cdots, X_n)$ with X_i being i.i.d. copies of X, then the code $c_n : \mathcal{X}^n \longrightarrow \mathcal{Y}^*$ satisfies $\mathbb{E}[|c_n(X_1, \cdots, X_n)|] < H_d(X_1, \cdots, X_n) + 1$. But $H_d(X_1, \cdots, X_n) = nH(X)$, hence

$$\frac{1}{n}\mathbb{E}[|c_n(X_1, \cdots, X_n)|] < H_d(X) + \frac{1}{n} \to H_d(X) \text{ as } n \to +\infty.$$

Put differently, one needs at least $H_d(X)$ symbols to encode one symbol in the source and this bound is attainable (at least asymptotically, using block codes).

To do this more explicitly, we recall the AEP. For large n, if we treat strings by breaking them into 'blocks' in $\mathcal{X}^n$, the AEP tells us which strings are 'typical'. This allows an efficient encoding: we associate short codewords with sequences in the typical set, and long codewords with the remaining atypical sequence.

Theorem 3.4. *(Source coding 1, Shannon's first theorem (1948)). Let $X_1, X_2, ...$ be an i.i.d. sequence of discrete random variables with state space $\mathcal{X}$. For every $\varepsilon > 0$, there exists an integer n, and a map*

$$c : \mathcal{X}^n \longrightarrow \{0,1\}^*$$

such that

(1) the induced map $\cup_{k\geq 0}\mathcal{X}^{nk} \longrightarrow \{0,1\}^*$ given by

$$(x_1, \cdots, x_{nk}) \mapsto c(\underbrace{x_1, ..., x_n}_{\in\mathcal{X}^n})c(\underbrace{x_{n+1}, ..., x_{2n}}_{\in\mathcal{X}^n}) \cdots c(\underbrace{x_{(k-1)n+1}, ..., x_{nk}}_{\in\mathcal{X}^n}) \in \{0,1\}^*$$

is injective;

(2) $\frac{1}{n}\mathbb{E}\big[|c(X_1, \cdots, X_n)|\big] \leq H(X) + \varepsilon$.

Proof. For some $\varepsilon_0 > 0$, we split $\mathcal{X}^n$ into the disjoint sets $\mathcal{T}_n^{\varepsilon_0}$ and $\mathcal{X}^n \backslash \mathcal{T}_n^{\varepsilon_0}$, and order the elements in $\mathcal{T}_n^{\varepsilon_0}$ and $\mathcal{X}^n \backslash \mathcal{T}_n^{\varepsilon_0}$ (in some arbitrary order; e.g. lexicographic). By the AEP, there are at most $2^{n(H(X)+\varepsilon_0)}$ elements in $\mathcal{T}_n^{\varepsilon_0}$, hence we can associate with every element of $\mathcal{T}_n^{\varepsilon_0}$ a string consisting of $l_1 := \lceil n(H(X) + \varepsilon_0)\rceil$ bits; similarly we associate with every element of $\mathcal{X}^n \backslash \mathcal{T}_n^{\varepsilon_0}$ a unique string of $l_2 = \lceil n \log(|\mathcal{X}|)\rceil$ bits. Now define $c(x_1, \cdots, x_n)$ as these strings with length l_1 resp. l_2 bits, prefixed by a 0 if $(x_1, \cdots, x_n)$ is in $\mathcal{T}_n^{\varepsilon_0}$, and prefixed by 1 otherwise. Clearly, this is injective (hence a bijection on

its image) and the prefix 0 or 1 indicates how many bits follow. This block code has expected length

$$\begin{aligned}
& \mathbb{E}[|c(X_1,\cdots,X_n)|] \\
= & \sum_{x\in\mathcal{T}_n^{\varepsilon_0}} p(x)(l_1+1) + \sum_{x\notin\mathcal{T}_n^{\varepsilon_0}} p(x)(l_2+1) \\
\leq & \sum_{x\in\mathcal{T}_n^{\varepsilon_0}} p(x)(n(H(X)+\varepsilon_0)+2) + \sum_{x\notin\mathcal{T}_n^{\varepsilon_0}} p(x)(n\log(|\mathcal{X}|)+2)) \\
\leq & \mathbb{P}((X_1,\cdots,X_n)\in\mathcal{T}_n^{\varepsilon_0})(n(H(X)+\epsilon_0)+2) + \mathbb{P}((X_1,\cdots,X_n)\notin\mathcal{T}_n^{\varepsilon_0})(n\log(|\mathcal{X}|)+2) \\
\leq & n(H(X)+\varepsilon_0)+2+\varepsilon_0 n\log(|\mathcal{X}|) \\
= & n(H(X)+\varepsilon_1)
\end{aligned}$$

with $\varepsilon_1 := \varepsilon_0(1+\log(|\mathcal{X}|)) + \frac{2}{n}$. For a given $\varepsilon > 0$, we first choose ε_0 small enough such that $\varepsilon_0(1+\log(|\mathcal{X}|)) < \varepsilon/2$, and then n sufficiently large such that $\frac{2}{n} \leq \varepsilon/2$. □

Shannon's first theorem shows that by focusing on typical sequences, we can encode a sequence $X_1, ..., X_n$ using on average no more than $nH(X)$ bits. Put differently: on average we need $H(X)$ bits to encode one symbol from this sequence. We will prove later that the above bound is sharp. This leads to another, more operational interpretation of the entropy of a random variable, namely as a compression bound on messages that are generated by i.i.d. sampling.

Remark 3.5. (1) The construction above only works for source sequences in $\cup_{k\geq 0}\mathcal{X}^{nk} \subsetneq \mathcal{X}^*$ (i.e. sequences of lengths divisible by n). This is usually not a concern, as given any sequence in $\mathcal{X}^*$, we can extend it to a sequence in $\cup_{k\geq 0}\mathcal{X}^{nk}$ by 'padding' it on the right with some repeated symbol from $\mathcal{X}$.

(2) The assumption that the sequence is generated by i.i.d. draws from the same distribution is not usually realistic (e.g. a sentence seen as sequences of letters, etc). However, this assumption can be significantly weakened and this is the content of the Shannon–McMillan–Breiman Theorem which we discuss in Chapter 5.

3.3 Shannon's code (1948)

The AEP argument in Theorem 3.4 is elegant, but it still does not give us a practical code unless we want to enumerate the typical sequences. In view of Theorem 3.2, a natural approach to construct a code is to assign to $x \in \mathcal{X}$ a codeword of length $\lceil -\log(p_X(x))\rceil$. Shannon gave an explicit algorithm that does this in his seminal 1948 paper: given a pmf p on $\mathcal{X} = \{1,\cdots,m\}, p_i = p(x_i)$, and a finite set $\mathcal{Y}$

(1) Write the probabilities p_i in decreasing order and assume (by relabelling) that $p_1 \geq \cdots \geq p_m$,

(2) Define the code $c_S(x_r)$ for x_r to be the first $l_r := \lceil -\log_{|\mathcal{Y}|}(p_r)\rceil$ digits in the $|\mathcal{Y}|$-ary expansion of the real number $\alpha_r := \sum_{i=1}^{r-1} p_i$.

The above construction is the 'Shannon code' c_S. Observe that the real numbers α_r and α_{r+1} differ by $p_r > |\mathcal{Y}|^{l_r} \geq |\mathcal{Y}|^{l_{r+1}}$, so following the proof of Theorem 2.5, one verifies that this is indeed a prefix code. As in Proposition 3.3, we also see that $H_{|\mathcal{Y}|}(X) \leq \mathbb{E}[|c_S(X)|] < H_{|\mathcal{Y}|}(X) + 1$. However,

- the Shannon code is in general not optimal,

- ordering a set of cardinality k needs $O(k\log(k))$ computational steps. This becomes prohibitively expensive when combined with the above block coding trick where we need to order $|\mathcal{X}|^n$ probabilities if we use blocks of length n; for example, already for uppercase English letters $\mathcal{X} = \{A, B, \cdots, Z\}$, using blocks of length $n = 100$, $|\mathcal{X}|^{100} = 26^{100}$ would require us to order and store(!) a set that contains more elements than there are particles in the known universe.

The Shannon code is highly dependent on the distribution of X. In practice, we usually have to infer the underlying probability distribution and work in a two step approach: firstly, read the whole message to infer the distribution; secondly, use the estimated pmf p to construct a code. The first step leads to errors, hence we need to ask how robust Shannon codes are.

Proposition 3.6. *Let p and q be pmf's on $\mathcal{X}$ and $X \sim p$ and $\mathcal{Y}$ a finite set of cardinality $|\mathcal{Y}| = d$. If we denote by $c_q : \mathcal{X} \longrightarrow \mathcal{Y}^*$ a Shannon code for the distribution q, then*

$$H_d(X) + D_d(p\|q) \leq \mathbb{E}[|c_q(X)|] < H_d(X) + D_d(p\|q) + 1.$$

Proof. We have

$$\begin{aligned}
\mathbb{E}[|c_q(X)|] &= \sum_{x\in\mathcal{X}} p(x)\lceil -\log_d(q(x))\rceil \\
&< \sum_{x\in\mathcal{X}} p(x)(-\log_d(q(x)) + 1) \\
&= \sum_{x\in\mathcal{X}} p(x)\left(\log_d\left(\frac{p(x)}{q(x)}\frac{1}{p(x)}\right) + 1\right) \\
&= \sum_{x\in\mathcal{X}} p(x)\left(\log_d\left(\frac{p(x)}{q(x)}\right)\right) + \sum_{x\in\mathcal{X}} p(x)\log_d\left(\frac{1}{p(x)}\right) + 1 \\
&= D_d(p\|q) + H_d(X) + 1.
\end{aligned}$$

Since the lower bound is attained iff $\lceil -\log_d(q(x))\rceil = -\log_d(q(x))$ the lower bound follows similarly. □

3.4 Fano's code (1949) (not examinable)

Fano suggested a different construction that is also very simple to implement. Given a pmf p on $\mathcal{X} = \{1, \cdots, m\}$ with $X \sim p$ and $p_i = p(x_i)$, and a finite set $\mathcal{Y}$ with $d = |\mathcal{Y}|$, Fano gave an explicit construction for a d-ary prefix code. In the case of a binary encoding the construction is as follows:

(1) Sort and relabel the symbols so that $p_1 \geq \cdots \geq p_m$;

(2) Find r that minimises $|\sum_{i\leq r} p_i - \sum_{i>r} p_i|$ and split $\mathcal{X}$ into two groups $\mathcal{X}_0 := \{x_i : i \leq r\}$ and $\mathcal{X}_1 := \{x_i : i > r\}$;

(3) Define the first digit of the codewords for $\mathcal{X}_0$ as 0 and for $\mathcal{X}_1$ as 1,

(4) Repeat Steps (2) and (3) recursively until we can not split anymore.

The above construction leads to the 'Fano code' (also called the Shannon–Fano code) $c_F : \mathcal{X} \longrightarrow \mathcal{Y}^*$. As for the Shannon code, it can be shown that $\mathbb{E}[|c_F(X)|] \leq H_d(X) + 2$, that the Fano code is a prefix code, and that in general the Fano code is not optimal.

3.5 Huffman codes (1952)

Huffman was a doctoral student who realised that prefix codes correspond to certain graphs, called rooted trees, and that previous constructions such as Fano's code build a tree starting at its root. As Huffman showed in 1952, by starting instead at the leaves of the tree, one gets a very simple algorithm that turns out to produce an optimal code!

Definition 3.7. *An undirected graph (V, E) is a tuple consisting of a set of vertices V and a set of edges $E = \{(v_1, v_2), v_1, v_2 \in V, v_1 \neq v_2\}$ a subset of $V \times V$. For $v \in V$ we denote by $\deg(v)$ the number of edges that contain v and call $\deg(v)$ the degree of v. We call a graph d-ary if the maximal degree of its vertices is d.*

We now define a subset of the set of graphs.

Definition 3.8. *The set of rooted trees $\mathcal{T}$ is a subset of all graphs and defined recursively as:*

(1) The graph τ consisting of a single vertex r is a rooted tree. We call r the root and the leaf of τ.

(2) If $\tau_i \in \mathcal{T}$ for $i = 1, \cdots, n$, then the graph τ formed by starting with a new vertex r and adding edges to each of the roots of $\tau_1, \cdots, \tau_n$ is also a rooted tree. We call r the root of τ and we call the leaves of $\tau_1, \cdots, \tau_n$ the leaves of τ.

We say a rooted tree is d-ary if each vertex has at most d edges which are leading away from the root. We say a d-ary rooted tree is labelled if each of the edges is assigned a number from $\{1, ..., d\}$, such that no two edges from the same vertex leading away from the root have the same number.

We can think of the set of prefix codes as the set of labelled rooted trees: identify the codewords with leaves, the empty message with the root vertex and labelling the edges by letters that are in the codeword at the leaf it ends up.

Lemma 3.9. *There is a bijection from the set of d-ary prefix codes to the set of labelled d-ary rooted trees.*

As remarked in Section 3.1, to find a prefix code with minimal expected length is an integer programming problem. Surprisingly, there exists a simple algorithm that constructs the prefix code of shortest expected length for a given distribution with linear complexity. This is the so-called Huffman code: we construct a rooted tree starting from the vertices of the least likely letters. For brevity of presentation, we describe only the binary Huffman code in detail: fix a pmf p on $\mathcal{X} = \{1, \cdots, m\}$ and a random variable $X \sim p$, and assume (by relabelling) that $p_1 \geq \cdots \geq p_m$ with $p_i := p(x_i)$. Then

(1) associate with the two least likely symbols, two leaves that are joined into a vertex;

(2) build a new distribution on $m-1$ symbols p, where $p'_1 = p_1, \cdots, p'_{m-2} = p_{m-2}$ and $p'_{m-1} := p_{m-1} + p_m$ (i.e. symbols $m-1$ and m are merged into one new symbol with probability $p'_{m-1} = p_{m-1} + p_m$), and relabel the resulting pmf in non-increasing order;

(3) repeat the above two steps of merging the two least likely symbols until we have a rooted tree.

Note that

- The algorithm can be seen as constructing the codetree from the bottom up: Step 2 amounts to joining two leaves with a new vertex.
- The algorithm terminates in $|\mathcal{X}|-1$ steps and once we have built the rooted tree the code assignment is done by assigning 0 or 1 to the branches. Hence the complexity is $O(|\mathcal{X}|)$ if we are given a sorted pmf p; if we need to sort the pmf then the complexity of constructing the Huffman code is $O(|\mathcal{X}| \log |\mathcal{X}|)$.
- If two symbols have same probability at every iteration, the resulting Huffman code may not be unique. However, any resulting codes will have the same expected length.
- In the d-ary case, the construction is analogous: we merge d vertices at every step. It may happen that we need to introduce dummy variables since there might not be enough vertices to merge d vertices. See [1] for details.

Proposition 3.10. *Let $\mathcal{X}, \mathcal{Y}$ be finite sets and p a pmf on $\mathcal{X}$ with a random variable $X \sim p$. The Huffman code $c : \mathcal{X} \longrightarrow \mathcal{Y}^*$ for p is optimal, i.e. if c' is another uniquely decodable code $c' : \mathcal{X} \longrightarrow \mathcal{Y}^*$, then*

$$\mathbb{E}[|c(X)|] \leq \mathbb{E}[|c'(X)|].$$

We prepare the proof with a Lemma about the general properties of a certain optimal prefix code. In itself it is not an important code but it is a useful tool to prove optimality of other codes (such as Huffman, as we will see in the proof Proposition 3.10).

Lemma 3.11. *Let p be a pmf on $\mathcal{X} = \{x_1, \cdots, x_m\}$ and assume wlog that $p_1 \geq \cdots \geq p_m$ for $p_i := p(x_i)$. Then there is a prefix code which is optimal, and any optimal prefix code c satisfies:*

(1) $p_j > p_k$ implies $|c(x_j)| \leq |c(x_k)|$,

(2) there are at least two longest codewords (with the same length),

(3) longest codewords come in pairs, which differ only in the last digit.

We call a code c with these properties a canonical code for the pmf p.

Proof. The existence of an optimal (prefix) code is as in Proposition 3.3. For Point (1), fix an optimal prefix code c and consider the code c' given by interchanging the codewords of c for x_j and x_k for some j, k with $j < k$ resp. $p_k < p_j$. Then

$$\begin{aligned} 0 &\leq \sum_i p_i|c'(x_i)| - \sum_i p_i|c(x_i)| \\ &= p_j|c(x_k)| + p_k|c(x_j)| - p_j|c(x_j)| - p_k|c(x_k)| \\ &= (p_j - p_k)(|c(x_k)| - |c(x_j)|). \end{aligned}$$

Hence $|c(x_k)| \geq |c(x_j)|$.

For Point (2), assume the contrary and remove the last digit from the longest codeword. This would still give a prefix code and this new prefix code would have strictly smaller expected length. Hence, there must be at least two longest codewords.

For Point (3), identify a prefix code with a rooted tree. A codeword of maximum length must have a sibling (a leaf connecting to same vertex; otherwise, we could remove the last digit and get a prefix code of shorter expected length). □

We now use this to prove that the Huffman code is optimal.

Proof of Proposition 3.10. Fix a pmf p with $p_1 \geq \cdots \geq p_m$ on m symbols. Denote by p' the pmf on $m-1$ symbols given by merging the lowest probabilities, $p'_i = p_i$ for $i \in \{1, \cdots, m-2\}$ and $p'_{m-1} = p_{m-1} + p_m$. Let c^p be a canonical optimal code for p. Define $c^{p'}$ as the code for p' given by merging the leaves for p_{m-1} and p_m in the rooted tree representing c^p (by Lemma 3.11, p_{m-1}, p_m are siblings so this is possible). Let $L(c) = \sum_i p_i |c(x_i)|$ denote the expected length of a code c. Then the difference in expected lengths between c^p and $c^{p'}$ is

$$\begin{aligned} L(c^p) - L(c^{p'}) &= p_{m-1}l + p_m l - p'_{m-1}(l-1) && (3.5.1) \\ &= p_{m-1} + p_m. && (3.5.2) \end{aligned}$$

where l denotes the codeword lengths of symbols $m-1$ and m under c^p. On the other hand, let $e^{p'}$ be any optimal (prefix) code for p'. We again represent it as a rooted tree and define e^p by replacing the leaf for p'_{m-1} with a rooted tree consisting of two leaves p_m and p_{m-1}. Then

$$L(e^p) - L(e^{p'}) = p_{m-1} + p_m. \tag{3.5.3}$$

Substracting (3.5.1) from (3.5.3) yields

$$(L(e^p) - L(c^p)) + (L(c^{p'}) - L(e^{p'})) = 0.$$

By assumption, c^p and $e^{p'}$ are optimal, hence both terms are non-negative so both must equal 0. We conclude that $L(e^p) = L(c^p)$, hence e^p is an optimal code for p. The above shows, that expanding any optimal code e' for p' leads to an optimal code e^p for p. Now note that each stage of the Huffman code is constructed by such an expansion. Further, for $m = 2$ the Huffman code is clearly optimal, hence the result follows by induction on m. □

The Huffman code has a simple construction and is optimal. It is used in mainstream compression formats (such as gzip, jpeg, mp3, png, etc). However, it is not the final answer to source coding.

- Not every optimal code is Huffman; for example the code in the following table is optimal but not

$p(x)$	0.3	0.3	0.2	0.2
$c(x)$	00	10	01	11

  Huffman (since c can be obtained by permuting leaves of the same length of the Huffman code for p).

- Huffman (and all the other prefix codes we have discussed so far) requires the ordering of the probabilities p_i, and we need to construct the full codebook in order to decode any message. Further, optimality was defined for messages that are drawn by taking i.i.d. samples. When compressing text (source symbols are English letters) this does not apply since, for instance, the probability of sampling e is much higher if the previous two letters were "th" compared with say "xy".
- Optimality just guarantees $H_d(X) \leq \mathbb{E}[|c(X)|] < H_d(X) + 1$. This is a good bound if $H_d(X)$ is large but for small entropies the term $+1$ on the right hand side is dominant. One can again use the block coding trick discussed in Section 3.2 to encode sequences of length n to reduce the overhead to $1/n$ bits but this again leads to a combinatorial explosion, since we need to sort $|\mathcal{X}|^n$ probability masses.

3.6 Elias' code

We are given a pmf p on $\mathcal{X} = \{1, \cdots, m\}$ with $p_i = p(x_i)$ and $X \sim p$, and a set $\mathcal{Y}$ of cardinality d. We associate the symbol x_r with the real number $\sum_{i<r} p_i + \frac{p_r}{2}$. In particular, the number associated with x_r differs from the number associated with any other source symbol x_j by at least $p_r/2$, and in (at most) the $\lceil -\log_d(p_r) \rceil + 1$ digit in its d-ary expansion. Using this observation, we define the Elias code (also Shannon–Fano–Elias code) $c_E(x_r)$ as the first $\lceil -\log_d(p_r) \rceil + 1$ digits in the d-ary expansion of the associated real number.

As above, one can show that $H_d(X) + 1 \leq \mathbb{E}[|c_E(X)|] < H_d(X) + 2$. Although it is less efficient than the above codes, this construction has the big advantage that we do not need to order the elements of $\mathcal{X}$ by their probabilities. Provided we know the probabilities p_i (in order), we also don't need to store the codebook, which is needed in the Huffman construction (as there may be more than one Huffman code!). It also leads to the more general 'arithmetic codes', which can achieve efficiency.

3.7 Arithmetic codes (1970s)

Arithmetic codes are another class of block codes, which have a simple construction, and give (asymptotically) optimal compression. They were discovered multiple times, including in Elias' unpublished work, Rissanen (1976) and Pasco (1976).

The aim is to split up the interval $[0, 1]$ into sections associated with each codeword. We assume $\mathcal{X} = \{1, ..., m\}$ and $\mathcal{Y} = \{1, .., d\}$, and write $F(x) = \sum_{k \leq x} p_k$ for the cumulative distribution function. We then associate each codeword x with the interval $[F(x-1), F(x))$, and note that this interval has length p_x. Elias' code reduces this interval to its midpoint and encodes that value with a length $\lceil -\log_d(p_r) \rceil + 1$ codeword, which gives a prefix code uniquely identifying each interval.

Arithmetic coding goes further, by repeating this with blocks, using a lexicographic order. We could simply list all blocks and apply the above method, but there is a simpler recursive interpretation. In order to encode a block x_1x_2, we first identify the initial section $[F(x_1 - 1), F(x_1))$. We now use this interval in the place of $[0, 1]$ above. If our source symbols are independent, this gives the interval

$$\Big[F(x_1-1) + F(x_2-1)\big(F(x_1) - F(x_1-1)\big), \quad F(x_1-1) + F(x_2)\big(F(x_1) - F(x_1-1)\big)\Big).$$

If our source symbols are not independent (for example, if they come from a Markov chain), then we can replace $F(x_2)$ by the corresponding *conditional* cumulative distribution function.

Decoding an arithmetic code is conceptually easy – we simply identify the interval, and find the codeword that corresponds to it. In practice, various tricks are used to avoid problems with variable-precision arithmetic, and so a good implementation is not completely straightforward.

Advantages of arithmetic coding is that it is a prefix code *within blocks* – you can extract the first character early, as you only need to identify that your message is in the interval $[F(x_1 - 1), F(x_1))$. However, codeword length is 'not wasted', as it gets used in the encoding of the subsequent character. For independent source symbols, we also do not need to store a codebook, as it can easily be reconstructed given the single-symbol probabilities and ordering. We can also use it to encode arbitrarily long messages, rather than blocks of fixed length (but to do this, we need to include an 'end of message' symbol in our source alphabet, or know the length of our original message, otherwise our codeword implicitly ends with an infinite string of x_1s).

You can also order the blocks in a convenient way, so that similar messages are close together (which gives the encoded message an interpretation as getting closer to the meaning of the source), or are far apart (so that errors in communication should be easily determined by context).

To prove optimality of arithmetic codes for independent symbols, we have the following result.

Theorem 3.12. *Let $X_1, ..., X_n$ be iid random variables from an alphabet $\mathcal{X}$, with pmf p. Then the arithmetic code based on length n blocks has average codeword length bounded above by $H(X_1, ..., X_n) + 2$, and hence per-character average codeword length at most $H(X) + 2/n$.*

Proof. A particular source block $x_1x_2...x_n$ will occur, due to independence, with probability $\prod_{i\leq n} p(x_i)$, which is (by construction), the length of the interval it is associated with. We encode the midpoint of this block using a string of length

$$\Big\lceil -\log\big(\prod_{i\leq n} p(x_i)\big)\Big\rceil + 1 \leq -\log\big(\prod_{i\leq n} p(x_i)\big) + 2.$$

The average block codeword length is then bounded by

$$-\sum_{x_1,...,x_n} \big(\prod_{i\leq n} p(x_i)\big) \log\big(\prod_{i\leq n} p(x_i)\big) + 2 = H(X_1, ..., X_n) + 2 = nH(X) + 2$$

and dividing by n gives the per-character length. □

A key advantage of arithmetic coding is its flexibility – at each stage you can change the probabilities. This means we can use Arithmetic coding to build optimal codes for Markov chains (where the probability depends on the preceding character) or in an adaptive way, where we change the probability distribution at each stage depending on the characters we have observed.

3.8 Variable-to-fixed codes

The codes we have been considering up until now are fixed-to-variable codes – that is, they take a fixed length input (in $\mathcal{X}^n$) and return a variable length output (in $\mathcal{Y}^*$). For some applications, it's convenient

to work the other way, and take a variable length input to a fixed length output. This is useful as it means that errors can only affect the current block, and cannot have long-run impact on the coded message. We will briefly consider two possible methods which do this.

We will focus on binary ($|\mathcal{Y}| = 2$) codes with $\mathcal{X} = \{1, ..., m\}$. If our output is of length d, then we know there are 2^d output messages available. Our aim is to find a set of input strings $\mathcal{W} \subset \mathcal{X}^*$ such that $|\mathcal{W}| = K \leq 2^d$, and which is *proper* (no element of $\mathcal{W}$ is a prefix of another element of $\mathcal{W}$) and *complete* (or *exhaustive*, i.e. every string in $\mathcal{X}^*$ has a prefix in $\mathcal{W}$). If we can do this with $K = 2^d$, then every output codeword is used.

3.8.1 Tunstall's code (1967)

Tunstall's code is similar in some senses to Fano's code. We build an m-ary tree. We start with a root and begin by assigning a leaf for each symbol in $\mathcal{X}$. We then find the leaf corresponding to the highest probability symbol and make it a root for m new leaves (so length two strings). This gives a tree with $2m-1$ leaves. Now repeat by finding the leaf corresponding to the highest probability string and splitting it into m leaves. After k iterations we have a tree with $k(m-1)+m$ leaves. We can do $k = \lfloor \frac{2^d - m}{m-1} \rfloor$ iterations until we run out of codewords. Now assign a codeword to each leaf according to some rule.

Example 3.13. Suppose $d = 3$ and $\mathcal{X} = \{a, b\}$ with $\mathbb{P}(X = a) = 0.7$. Then an example of Tunstall's iteration is

Step									
1	Symbol	a	b	-	-	-	-	-	-
	Prob	0.7	0.3	-	-	-	-	-	-
2	Symbol	aa	b	ab	-	-	-	-	-
	Prob	0.49	0.3	0.21	-	-	-	-	-
3	Symbol	aaa	b	ab	aab	-	-	-	-
	Prob	0.343	0.3	0.21	0.147	-	-	-	
4	Symbol	aaaa	b	ab	aab	aaab	-	-	-
	Prob	0.2401	0.3	0.21	0.147	0.1029	-	-	-
5	Symbol	aaaa	ba	ab	aab	aaab	bb	-	-
	Prob	0.2401	0.21	0.21	0.147	0.1029	0.09	-	-
6	Symbol	aaaaa	ba	ab	aab	aaab	bb	aaaab	-
	Prob	0.16807	0.21	0.21	0.147	0.1029	0.09	0.07203	
7	Symbol	aaaaa	baa	ab	aab	aaab	bb	aaaab	bab
	Prob	0.16807	0.147	0.21	0.147	0.1029	0.09	0.07203	0.063
	Encoding	000	001	010	011	100	101	110	111

This achieves a compression of, on average, 3.2831 characters per length-3 block. Using this, the sequence aaababaaaaa would be encoded as 100 010 000. As with block codes, we may need to 'pad' the input message in order to get a valid input.

We can also give an optimality result for Tunstall's code.

Theorem 3.14. *Suppose we have an i.i.d. sequence $\{X_i\}_{i=1}^{\infty}$ of inputs taking values in $\mathcal{X}$. Among*

variable-to-fixed codes, Tunstall's code is optimal, in the sense that it maximizes the expected length of the input string which is encoded in a single output in $\mathcal{Y}^d$, among all complete and proper codes.

Proof. (Sketch) We generalize from the case where there are 2^d output messages, by allowing a generic (integer) number of outputs. Any complete and proper variable-to-fixed code must correspond to an m-ary tree for inputs (analogously to what we saw for Huffman's code).

We know that every m-ary tree has $k(m-1)+m$ leaves, for some integer k (the number of non-root internal vertices in the tree). Let $\mathcal{K}(T)$ be the set of non-root internal vertices in a tree T. We then see that the expected input message length of a proper code is $1 + \sum_{n \in \mathcal{K}(T)} P(n)$, where $P(n)$ is the probability of the vertex.

To select an optimal code, we need to maximize the values of $P(n)$. We view T as a subtree of the infinite m-ary tree T_∞, which is built by selecting $|\mathcal{K}|$ internal vertices. These vertices need to build a tree, but if we ignore that fact, we can just ask about arbitrarily selecting $|\mathcal{K}|$ vertices.

As the descendants of any vertex will always have smaller probability than the parent, we can see that Tunstall's code is precisely selecting the maximal probability vertices. This tells us that Tunstall's code is optimal (and that the maximal probability vertices always form a complete code). □

There are various additional results that can be shown, for example, if L is the length of a random sequence encoded as one output, then $d/\mathbb{E}[L] \geq H(X)$, in agreement with the bounds we have seen for fixed-to-variable codes.

Alternative variable-to-fixed methods include the Block Arithmetic Codes (which has advantages similar to arithmetic codes), due to Bonocelet (1993); and the Lempel–Ziv (1977/78) and Lempel–Ziv–Welch (1984) methods (which are adaptive, in that they do not assume knowledge of the distribution of X).

Chapter 4

Channel Coding: Shannon's Second Theorem

In Chapters 2 and 3 we studied how much information is contained in sequences and used this to derive codes to store such sequences. In many real-world situations we are confronted with the problem of transmitting information from one place to another, typically through a medium that is subject to noise and perturbations.

4.1 Discrete memoryless channels

Definition 4.1. *A discrete memoryless channel (DMC) is a triple $(\mathcal{X}, M, \mathcal{Y})$ consisting of*

- *a finite set $\mathcal{X}$, called the input alphabet,*
- *a finite set $\mathcal{Y}$, called the output alphabet,*
- *a stochastic[1] $|\mathcal{X}| \times |\mathcal{Y}|$-matrix M, called the emission matrix.*

We say that a pair of random variables X, Y defined on some probability space $(\Omega, \mathcal{F}, \mathbb{P})$ realises the DMC, if the conditional distribution of Y given X equals M, i.e. $M = \big(p_{Y|X}(y|x)\big)_{x\in\mathcal{X}, y\in\mathcal{Y}}$.

Example 4.2. $\mathcal{X} = \{0, 1\}$, $\mathcal{Y} = \{a, b, c, d, e\}$ and $M = \begin{pmatrix} 0.2 & 0 & 0.5 & 0 & 0.3 \\ 0 & 0.2 & 0 & 0.8 & 0 \end{pmatrix}$.

This is an example of a lossless channel: knowing the output $\mathcal{Y}$ allows us to uniquely identify the input X (e.g. the output is b or d iff the input is 1). More generally, we call $(\mathcal{X}, M, \mathcal{Y})$ a lossless channel if we can divide $\mathcal{Y}$ into disjoint sets $\mathcal{Y}_1, \cdots, \mathcal{Y}_{|X|}$ such that

$$\mathbb{P}(Y \in \mathcal{Y}_i | X = x_i) = 1 \text{ for } \forall\, 1 \leq i \leq |\mathcal{X}|.$$

For a lossless channel $(\mathcal{X}, M, \mathcal{Y})$, similar to Point (2) in Theorem 1.19, we have $H(X|Y) = 0$ (since $X = f(Y)$ for $f(y) = x_i \mathbf{1}_{y\in\mathcal{Y}_i}$, i.e., X is a deterministic function of Y).

[1] A stochastic matrix is a matrix with non-negative entries and the sum of entries in each row equals 1.

Another extreme is a channel that is completely useless for transmitting information, i.e. the output Y contains no information about the input X. This means X and Y are independent, which is (again by Point (2) in Theorem 1.19) equivalent to $H(X|Y) = H(X)$.

Here are some important examples of DMCs.

Example 4.3. Let $q \in [0,1]$.

(1) **Binary symmetric channel**: $\mathcal{X} = \mathcal{Y} = \{0,1\}$ and the stochastic matrix is given as

$\mathcal{X} \setminus \mathcal{Y}$	0	1
0	$1-q$	q
1	q	$1-q$

(2) **Binary erasure channel**: $\mathcal{X} = \{0,1\}$, $\mathcal{Y} = \{0,1,e\}$ and the stochastic matrix is given as

$\mathcal{X} \setminus \mathcal{Y}$	0	e	1
0	$1-q$	q	0
1	0	q	$1-q$

(3) **Noisy typewriter**: $\mathcal{X} = \mathcal{Y} = \{A, \cdots, Z\}$ and the stochastic matrix is given as

$\mathcal{X} \setminus \mathcal{Y}$	A	B	C	D	$\cdots$	$\cdots$	Y	Z
A	1/3	1/3	0	0	$\cdots$	$\cdots$	0	1/3
B	1/3	1/3	1/3	0	$\cdots$	$\cdots$	0	0
$\vdots$		$\ddots$		$\ddots$		$\ddots$		$\ddots$
Y	0	0	0	$\cdots$	0	1/3	1/3	1/3
Z	1/3	0	0	$\cdots$	0	0	1/3	1/3

4.2 Channel capacity

We want to measure how much our uncertainty about the input X is reduced by knowing the output Y. We have seen that a lossless channel $H(X|Y) = 0$ and a useless channel $H(X|Y) = H(X)$. Motivated by this, an intuitive measure for the quality of a channel is

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

A DMC only specifies the distribution of the output conditional on the input, that is, $p_{Y|X}$. To use the channel for information transmission, we have freedom to choose the distribution of the input. For each input distribution, p_X, we have the corresponding output distribution

$$p_Y(y) = \sum_x p_{Y|X}(y|x) p_X(x)$$

and joint distribution $p(x,y) = p_{Y|X}(y|x)p_X(x)$.

This motivates the definition of channel capacity.

Definition 4.4. *Let $(\mathcal{X}, M, \mathcal{Y})$ be a DMC. We call $C := \sup I(X;Y)$ the channel capacity of the DMC $(\mathcal{X}, M, \mathcal{Y})$, where the supremum is taken over all input distributions p_X.*

From $I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$, it follows by Theorem 1.19 that

$$0 \leq C \leq \min\{\log(|\mathcal{X}|), \log(|\mathcal{Y}|)\}.$$

These upper bounds are measuring 'how many bits of information are in a single input/output character', which is an intuitive bound on how much information a channel can transmit.

Proposition 4.5. *For fixed $p_{Y|X}$, the map $p_X \to I(X;Y)$ is concave. Conversely, for fixed p_X, the map $p_{Y|X} \to I(X;Y)$ is convex.*

Proof. For the first statement, recall $I(X;Y) = H(Y) - H(Y|X)$. Given $p_{Y|X}$, we know p_Y and $H(Y|X)$ are linear functions of p_X. We also know $H(Y)$ is concave in p_Y, so $H(Y)$ is concave in p_X, and $I(X;Y)$ is concave in p_X

For the second statement, recall $I(X;Y) = D(p_{X,Y} \| p_X p_Y)$. Given p_X, we know $p_{X,Y}$ is linear in $p_{Y|X}$. Taking the marginal distribution of Y shows $p_X p_Y$ is also linear in $p_{Y|X}$. By Point (5) in Theorem 1.16, we know $D(p\|q)$ is convex in (p,q). So $I(X;Y)$ is convex in $p_{Y|X}$ for any fixed p_X. □

Remark 4.6. As $p_X \mapsto I(X;Y)$ is concave and continuous it has a unique maximum, which is attained (as p_X takes values in a compact set). We can usually find this maximum using a simple first-order condition.

Below we calculate the capacity of some simple channels.

Example 4.7. For the binary symmetric channel specified in Example 4.3(1), we have a transmission error with probability q. To calculate its capacity, we need to find a tight upper bound on $I(X;Y)$.

$$\begin{aligned} I(X;Y) &= H(Y) - H(Y|X) \\ &= H(Y) - \sum_{x \in \mathcal{X}} p(x) H(Y|X=x) \\ &= H(Y) - \sum_{x \in \mathcal{X}} p(x) H(q) \end{aligned}$$

where $H(q) = -q\log(q) - (1-q)\log(1-q)$ is the entropy of the pmf $(q, 1-q)$ (Bernouli distribution). We could optimize this directly with respect to p (which implicitly appears in both terms, as it affects Y), but a clever estimate gives

$$I(X;Y) \leq 1 - H(q).$$

Firstly we note that, if Y is uniform, $\mathbb{P}(Y=0) = \mathbb{P}(Y=1) = 1/2$, then $H(Y) = 1$ and this is an equality. Now, since

$$\begin{aligned} p_Y(0) &= (1-q)p_X(0) + q p_X(1), \\ p_Y(1) &= q p_X(0) + (1-q)p_X(1), \end{aligned}$$

we know that $\mathbb{P}(Y=0)=1/2$ is equivalent to $\mathbb{P}(X=0)=1/2$ by the symmetry between the roles of X and Y. Hence the maximum is $C=1-H(q)$, which is attained iff $\mathbb{P}(X=0)=1/2$.

Example 4.8. In Example 4.3(2), a binary erasure channel is specified by $\mathcal{X}=\{0,1\}$ and $\mathcal{Y}=\{0,e,1\}$, where e can be interpreted as an error occurred in the transmission, and the stochastic matrix M given by

$\mathcal{X}\setminus\mathcal{Y}$	0	e	1
0	$1-q$	q	0
1	0	q	$1-q$

The binary channel erases a fraction q of the bits that are transmitted and the receiver knows which bits have been erased. Hence, we can only hope to recover $1-q$ bits in proportion. Now, as before, $I(X;Y)=H(Y)-H(Y|X)=H(Y)-H(q)$ with $H(q)$ the same as in the previous example. Set $\pi=\mathbb{P}(X=1)$, then $p_Y(0)=(1-\pi)(1-q)$, $p_Y(e)=(1-\pi)q+\pi q=q$, $p_Y(1)=\pi(1-q)$, so

$$\begin{aligned} H(Y) &= -(1-\pi)(1-q)\log((1-\pi)(1-q))-q\log(q)-\pi(1-q)\log(\pi(1-q)) \\ &= -(1-\pi)(1-q)\log(1-\pi)-(1-\pi)(1-q)\log(1-q)-q\log(q) \\ &\quad -\pi(1-q)\log(\pi)-\pi(1-q)\log(1-q) \\ &= H(q)+(1-q)H(\pi). \end{aligned}$$

Now

$$I(X;Y)=H(q)+(1-q)H(\pi)-H(q)=(1-q)H(\pi)$$

and therefore, maximizing over π, the capacity is $C=1-q$ achieved with $\pi=\mathbb{P}(X=1)=1/2$.

4.3 Channel codes, rates and errors

We want to use the channel to reliably transmit a message from a given set of possible messages. We are allowed to use the channel several times. Hence, we are looking for a map that transforms the message into a sequence of symbols in $\mathcal{X}$ (encoding), we then send this sequence through the channel and upon receiving the corresponding sequence symbols in $\mathcal{Y}$, transforms this back to a message (decoding) with a small probability of error. Note that this is not the same meaning of $\mathcal{X}$ as in the previous sections (where we had $\mathcal{X}$ as the space of original messages).

Definition 4.9. *Fix $m,n\geq 1$. A (m,n)-channel code for a DMC $(\mathcal{X},M,\mathcal{Y})$ is a tuple (c,d) consisting of*

- *a map $c:\{1,\cdots,m\}\longrightarrow\mathcal{X}^n$, called the encoder,*
- *a map $d:\mathcal{Y}^n\longrightarrow\{1,\cdots,m\}$, called the decoder.*

We call $\{1,\cdots,m\}$ the message set, $c(i)$ the codeword for message $i\in\{1,\cdots,m\}$ and the collection $\{c(i):i=1,\cdots,m\}$ the codebook.

That is to say, a (m, n) channel transmits one out of m messages by using the channel n times.

Definition 4.10. *Let $(\mathcal{X}, M, \mathcal{Y})$ be a DMC. We call $\rho(c, d) := \frac{1}{n}\log(m)$ the (binary) rate of the (m, n)-code (c, d).*

Remark 4.11. The logic of this definition is that a simple binary code can express our m messages in $\log(m)$ bits. Using c and d, we encode these m messages using a string of length n, so we are sending $\frac{1}{n}\log(m) = \rho(c, d)$ bits of information per character in the DMC.

Definition 4.12. *Let (c, d) be a (m, n)-channel code for a DMC $(\mathcal{X}, M, \mathcal{Y})$. Set*

$$\varepsilon_i = \mathbb{P}(d(\mathbf{Y}) \neq i \,|\, c(i) = \mathbf{X}) \text{ for } i = 1, \cdots, m,$$

where $\mathbf{X} = (X_1, \cdots, X_n)$ and $\mathbf{Y} = (Y_1, \cdots, Y_n)$ with $\{(X_i, Y_i)\}_{i=1,\cdots,n}$ consisting of i.i.d. copies of random variables (X, Y) that realise the DMC. We say that the channel code has

(1) a maximal probability of error $\varepsilon_{max} := \max_{i\in\{1,\cdots,m\}} \varepsilon_i$,

(2) an (arithmetic) average error $\bar{\varepsilon} := \frac{1}{m}\sum_{i=1}^{m} \varepsilon_i$.

Remark 4.13. For applications we clearly care about ε_{max} and *a priori* it is not clear that $\bar{\varepsilon}$ is a useful quantity to consider. Note that $\bar{\varepsilon} \leq \varepsilon_{max}$ and that $\bar{\varepsilon}$ is the expectation of the error ε_i, if an element i is chosen uniformly at random. It turns out that good estimates on $\bar{\varepsilon}$ imply good estimates on ε_{max} and that bounds on $\bar{\varepsilon}$ are easy to establish (we are going to use this in the proof of the noisy channel coding theorem).

Already a simple repetition code (represent the message i in its $|\mathcal{X}|$-ary expansion and transmit each digit multiple times) can achieve an arbitrary small error at the cost of a vanishing rate. We therefore need to understand the tradeoff between the error probability ε_{max} (which we want to make small) and the rate R (which we want to keep large). That is, we ask what points in the (ε_{max}, R)-plane can be reached by channel codes (with a sufficiently large n)? Before Shannon, a common belief was that that as ε_{max} goes to 0 so does the rate. A big surprise was Shannon's noisy channel coding theorem, that showed that any rate below channel capacity can be achieved!

4.4 Shannon's second theorem: noisy channel coding

Definition 4.14. *A rate $R > 0$ is achievable for a DMC $(\mathcal{X}, M, \mathcal{Y})$ if, for any $\varepsilon > 0$, there exists sufficiently large m, n and an (m, n)-channel code (c, d) with*

$$\rho(c, d) > R - \varepsilon \text{ and } \varepsilon_{max} < \varepsilon,$$

where ε_{max} denotes the maximal error of (c, d).

In other words, a rate R is achievable if there exists a sequence of codes whose rates approach R and whose maximal errors when sending messages through the DMC approach zero. *A priori* it is by no means obvious that a message may be transmitted over a DMC at a given rate with as small a probability of error as desired! Shannon's result not only shows that this is possible but also shows that the set of

rates that can be achieved (in theory) is exactly those that are bounded by the channel capacity C. We already saw that the channel capacity can be explicitly computed for some important channels. All these are reasons why Theorem 4.15 is considered a (maybe even the) major result of communication theory.

Theorem 4.15. *(Shannon's second theorem: noisy channel coding). Let $(\mathcal{X}, M, \mathcal{Y})$ be a DMC with capacity C. Then a rate $R > 0$ is achievable iff $R \leq C$.*

An analogy that is often used is to compare a channel to a water pipe. If we pump water through a pipe above its capacity, then the pipe will burst and water will be lost. Similarly, if information flows through a channel at rate higher than channel capacity, the error is strictly bounded away from zero which means we lose information.

Let us first give an informal "proof" of Shannon's channel coding theorem. The idea is to use a "typical set decoder": define a decoder by partitioning $\mathcal{Y}^n$ into disjoint subsets $\mathcal{Y}_1, \cdots, \mathcal{Y}_m$ of $\mathcal{Y}^n$, and associate each set with an input sequence $x_1, \cdots, x_m \in \mathcal{X}^n$. That is, upon a receiving a sequence $y \in \mathcal{Y}^n$, if we find an i such that $y \in \mathcal{Y}_i$, then we decode it as message i. How can we find a partition that is efficient and robust to the noise in the channel? The key insight is similar to source coding: sequences can be divided into a set of typical sequences that carries most of the probability mass. There are approximately $2^{nH(Y)}$ typical output sequences. Similarly, to a given typical input sequence x correspond approximately $2^{nH(Y|X)}$ output sequences that are likely (i.e. y's such that (x, y) is typical with respect to $p_{X,Y}$). But for two different typical input sequences, these subsets of $\mathcal{Y}^n$ might overlap. To account for this we restrict ourselves further to a subset of typical input sequences such that the corresponding sets of typical output sequences do not overlap (but still cover nearly all of) $\mathcal{Y}^n$. There are at most

$$\frac{2^{nH(Y)}}{2^{nH(Y|X)}} = 2^{n(H(Y)-H(Y|X))} = 2^{nI(X;Y)}$$

such typical input sequences. Hence, there are at most $2^{nI(X;Y)}$ codewords, which gives a rate of $\frac{\log(2^{nI(X;Y)})}{n} = I(X;Y) \leq C$ bits per channel use. This shows (very heuristically) why we can expect to achieve any rate $R \leq C$.

Definition 4.16. *Let (X, Y) be a $\mathcal{X} \times \mathcal{Y}$-valued random variable with pmf $p_{X,Y}$. For $n \in \mathbb{N}$ and $\varepsilon > 0$, set $\mathbf{X} = (X_1, \cdots, X_n), \mathbf{Y} = (Y_1, \cdots, Y_n)$ with entries i.i.d. copies of X, Y, and*

$$\mathcal{J}_\varepsilon^{(n)} = \Bigg\{ (x,y) \in \mathcal{X}^n \times \mathcal{Y}^n : \max\Bigg(\left| \frac{-\log(p_{\mathbf{X},\mathbf{Y}}(x,y))}{n} - H(X,Y) \right|, \quad \left| \frac{-\log(p_{\mathbf{X}}(x))}{n} - H(X) \right|, \\ \left| \frac{-\log(p_{\mathbf{Y}}(y))}{n} - H(Y) \right| \Bigg) < \varepsilon \Bigg\}.$$

We call $\mathcal{J}_\varepsilon^{(n)}$ the set of jointly typical sequences of length n and tolerance ε.

Theorem 4.17. *(Joint AEP). Consider a sequence of pairs of random variables (X_i, Y_i) with $i = 1, ..., n$. We assume that (X_i, Y_i) and (X_j, Y_j) are i.i.d. for $i \neq j$ (but X_i and Y_i are generally not independent). We write $\mathbf{X} = (X_1, \cdots, X_n)$ and $\mathbf{Y} = (Y_1, \cdots, Y_n)$. Similarly, consider a sequence (X_i', Y_i') for $i = 1, ..., n$ which are i.i.d. in i, where X_i' and Y_i' have the same marginal distributions as X_i, Y_i, but X_i' and Y_i' are independent for all i. Write $\mathbf{X}'$ and $\mathbf{Y}'$ accordingly. Then, for $\varepsilon > 0$,*

(1) $\lim_{n \to +\infty} \mathbb{P}((\mathbf{X}, \mathbf{Y}) \in \mathcal{J}_\varepsilon^{(n)}) = 1$;

(2) For each $n \in \mathbb{N}$ we have $|\mathcal{J}_\varepsilon^{(n)}| \leq 2^{n(H(X,Y)+\varepsilon)}$;

(3) $\exists\, n_0$ such that $\forall\, n \geq n_0$, we have

$$(1-\varepsilon)2^{-n(I(X;Y)+3\varepsilon)} \leq \mathbb{P}\left((\mathbf{X}', \mathbf{Y}') \in \mathcal{J}_\varepsilon^{(n)}\right) \leq 2^{-n(I(X;Y)-3\varepsilon)},$$

where the upper bound holds for all $n \geq 1$.

Remark 4.18. Point (3) is the key new result, as it shows that if $I(X;Y) > 0$, then $\mathbf{X}$ and $\mathbf{Y}$ will not look like independent sequences for large n.

Proof. Point (1) follows by independence and the weak law of large numbers. As $\frac{\log(p(X_1,\cdots,X_n))}{n} = \frac{\sum_{i=1}^n \log(p(X_i))}{n} \to H(X)$ in probability, for any $\epsilon' > 0$,

$$\mathbb{P}\left(\left|\frac{-\log(p_{\mathbf{X}}(X_1,\cdots,X_n))}{n} - H(X)\right| \geq \varepsilon\right) < \frac{\varepsilon'}{3} \text{ for all } n \geq n_1$$

for some integer n_1, and similarly

$$\begin{aligned}
\mathbb{P}\left(\left|\frac{-\log(p_{\mathbf{Y}}(Y_1,\cdots,Y_n))}{n} - H(Y)\right| \geq \varepsilon\right) &< \frac{\varepsilon'}{3} \text{ for all } n \geq n_2, \\
\mathbb{P}\left(\left|\frac{-\log(p_{\mathbf{X},\mathbf{Y}}(X_1,\cdots,X_n,Y_1,\cdots,Y_n))}{n} - H(X,Y)\right| \geq \varepsilon\right) &< \frac{\varepsilon'}{3} \text{ for all } n \geq n_3.
\end{aligned}$$

for some integers n_2, n_3. Hence, taking $n \geq \max(n_1, n_2, n_3)$ we have

$$\mathbb{P}((\mathbf{X}, \mathbf{Y}) \notin \mathcal{J}_\varepsilon^{(n)}) \leq \varepsilon',$$

which gives the limit result.

Point (2) follows since for any $n \in \mathbb{N}$

$$1 = \sum_{\mathcal{X}^n \times \mathcal{Y}^n} p_{\mathbf{X},\mathbf{Y}}(x,y) \geq \sum_{\mathcal{J}_\varepsilon^{(n)}} p_{\mathbf{X},\mathbf{Y}}(x,y) \geq |\mathcal{J}_\varepsilon^{(n)}| 2^{-n(H(X,Y)+\varepsilon)},$$

and therefore $|\mathcal{J}_\varepsilon^{(n)}| \leq 2^{n(H(X,Y)+\varepsilon)}$.

Point (3) requires two bounds. For the upper bound,

$$\begin{aligned}
\mathbb{P}\left((\mathbf{X}', \mathbf{Y}') \in \mathcal{J}_\varepsilon^{(n)}\right) &= \sum_{\mathcal{J}_\varepsilon^{(n)}} p_{\mathbf{X}}(x) p_{\mathbf{Y}}(y) \\
&\leq 2^{n(H(X,Y)+\varepsilon)} 2^{-n(H(X)-\varepsilon)} 2^{-n(H(Y)-\varepsilon)} \\
&= 2^{-n(I(X;Y)-3\varepsilon)}.
\end{aligned}$$

For the lower bound, for large enough n we know that $\mathbb{P}\left((\mathbf{X}, \mathbf{Y}) \in \mathcal{J}_\varepsilon^{(n)}\right) \geq 1 - \varepsilon$, hence

$$1 - \varepsilon \leq \sum_{\mathcal{J}_\varepsilon^{(n)}} p_{\mathbf{X},\mathbf{Y}}(x,y) \leq \left|\mathcal{J}_\varepsilon^{(n)}\right| 2^{-n(H(X,Y)-\varepsilon)},$$

and we get $\left|\mathcal{J}_\varepsilon^{(n)}\right| \geq (1-\varepsilon)2^{n(H(X,Y)-\varepsilon)}$. Using this, in the same way as above,

$$\begin{aligned}
\mathbb{P}\left((\mathbf{X}', \mathbf{Y}') \in \mathcal{J}_\varepsilon^{(n)}\right) &= \sum_{\mathcal{J}_\varepsilon^{(n)}} p_{\mathbf{X}}(x) p_{\mathbf{Y}}(y) \\
&\geq (1-\varepsilon) 2^{n(H(X,Y)-\varepsilon)} 2^{-n(H(X)+\varepsilon)} 2^{-n(H(Y)+\varepsilon)} \\
&= (1-\varepsilon) 2^{-n(I(X,Y)+3\varepsilon)}.
\end{aligned}$$

□

We now use Theorem 4.17 to give a rigorous proof of Shannon's channel coding theorem.

Proof of Theorem 4.15.

(*Step 1: Rate is achievable.*) We want to find a code with rate close to $C = \sup_{p_X} I(X;Y)$. We fix $p = p_X$ to be a maximizer of $I(X;Y)$, some $\varepsilon \in (0, 3C/2)$, and choose m, n to be integers which are sufficiently large (we will see how large as part of the proof).

(*Step 1a: Defining a random code.*) Let $\mathcal{J}_\varepsilon^{(n)}$ be the jointly typical set of $p_{X,Y} = p_{Y|X}p_X$. We generate a random $(2m, n)$-channel code as follows:

(1) Generate $2m$ random codewords in $\mathcal{X}^n$, by sampling independently from $\Pi_{i=1}^n p_X(x_i)$;

(2) For each message $i \in \{1, \cdots, 2m\}$, define its encoding as the corresponding random codeword;

(3) Define the decoder as a typical-set decoder: upon receiving $\mathbf{Y}$, check if there exists a unique element $\mathbf{X}$ in the set of random codewords such that $(\mathbf{X}, \mathbf{Y}) \in \mathcal{J}_{\varepsilon/6}^{(n)}$. In this case, decode as the message that was in step (2) associated with the codeword $\mathbf{X}$. If this is not the case (there does not exist such a codeword or it is not unique) the decoder outputs $2m$.

Denote this random $(2m, n)$-channel code by $(\mathcal{C}, \mathcal{D})$.

(*Step 1b: Calculating the conditional probability of error*) We calculate the probability of error when we follow the sequence:

(1) Sample from the channel code $(\mathcal{C}, \mathcal{D})$;

(2) Sample a message W uniformly from $\{1, \cdots, 2m\}$;

(3) Send the sequence $\mathbf{X} = \mathcal{C}(W)$ through the channel;

(4) Decode the channel output using $\mathcal{D}$, denote the decoded message with $\hat{W}$.

Denote by E_i the event that the random codeword for i and the channel output are jointly typical (i.e. in $\mathcal{J}_{\varepsilon/6}^{(n)}$). By construction of the random code, ε_i is the same for all messages $i < 2m$, and is lower if $i = 2m$ (as this was the default output case). Therefore, $\varepsilon_{max} = \varepsilon_1$ for our random code.

We wish to calculate

$$\varepsilon_{max} = \varepsilon_1 = \mathbb{P}(\hat{W} \neq 1 | W = 1)$$

An error can occur either because we are in E_1^c (so we have output not jointly typical with our input message codeword) or are in E_i for some $i \neq 1$ (so we have output jointly typical with another codeword). By the union bound for probabilities

$$\mathbb{P}(\hat{W} \neq 1 | W = 1) = \mathbb{P}\left(E_1^c \cup (\cup_{i=2}^{2m} E_i) \,|\, W = 1\right) \leq \mathbb{P}(E_1^c \,|\, W = 1) + \sum_{i=2}^{2m} \mathbb{P}(E_i | W = 1).$$

By joint typicality, $\mathbb{P}(E_1^c | W = 1) < \frac{\varepsilon}{6}$. When $W = 1$, we can think of E_i (for $i \neq 1$) as the event that we have independently chosen X and Y and just happened to find a jointly typical sequence – hence

$\mathbb{P}(E_i|W=1) \leq 2^{-n(I(X;Y)-3\varepsilon/6)} = 2^{-n(I(X;Y)-\varepsilon/2)}$ for $i \neq 1$, provided n is sufficiently large. Combining these,

$$\begin{aligned} \varepsilon_{max} &\leq \frac{\varepsilon}{6} + \sum_{i=2}^{2m} 2^{-n(I(X;Y)-\varepsilon/2)} \\ &\leq \frac{\varepsilon}{6} + (2m-1)2^{-n(I(X;Y)-\varepsilon/2)} \\ &< \frac{\varepsilon}{6} + m2^{-n(I(X;Y)-\varepsilon/2)+1} \end{aligned}$$

We now choose n sufficiently large that these bounds hold, and also

$$n > \max\left\{\frac{1}{C-2\varepsilon/3}, -\frac{6}{\varepsilon}\log(\varepsilon/6)\right\} \qquad \text{and} \qquad m = \lfloor 2^{n(C-2\varepsilon/3)} \rfloor.$$

It follows that $m > 2^{n(C-2\varepsilon/3)-1}$, so the random $(2m, n)$-channel code $(\mathcal{C}, \mathcal{D})$ has rate

$$\frac{1}{n}\log(2m) \geq \frac{1}{n}n(C-2\varepsilon/3) = C - \frac{2\varepsilon}{3}.$$

We also know $m \leq 2^{n(C-2\varepsilon/3)}$ and $2^{1-n\varepsilon/6} < \varepsilon/3$, and that $I(X;Y) = C$, therefore

$$\varepsilon_{max} = \mathbb{P}(W \neq \hat{W}) \leq \frac{\varepsilon}{6} + 2^{n(C-2\varepsilon/3)}2^{-n(I(X;Y)-\varepsilon/2)+1} = \frac{\varepsilon}{6} + 2^{1-n\varepsilon/6} < \frac{\varepsilon}{2}.$$

(*Step 1c: Finding a deterministic code*) By conditioning, we see that our random code satisfies

$$\mathbb{P}(W \neq \hat{W}) = \sum_{(c,d)} \mathbb{P}(W \neq \hat{W}|(\mathcal{C}, \mathcal{D}) = (c,d))\mathbb{P}((\mathcal{C}, \mathcal{D}) = (c,d)) < \frac{\varepsilon}{2}.$$

It follows that there must exist at least one channel code (c^*, d^*) such that

$$\mathbb{P}(W \neq \hat{W}|(\mathcal{C}, \mathcal{D}) = (c^*, d^*)) < \frac{\varepsilon}{2}.$$

Recall that W was sampled uniformly and the average error is the expected error over all messages if the input is uniformly distributed. Hence, the above inequality can be restated as $\bar{\varepsilon} < \frac{\varepsilon}{2}$, where $\bar{\varepsilon}$ denotes the average error of (c^*, d^*). Thus we have shown the existence of a $(2m, n)$-channel code with rate at least $C - \frac{2\varepsilon}{3}$ and average error $\bar{\varepsilon} < \frac{\varepsilon}{2}$. Further,

$$\bar{\varepsilon} = \frac{1}{2m}\sum_{i=1}^{2m} \varepsilon_i < \frac{\varepsilon}{2},$$

or equivalently, $\sum_{i=1}^{2m} \varepsilon_i < m\varepsilon$ (here ε_i denotes the probability of an error in decoding message i using channel code (c^*, d^*)). Now sort the codewords by their error probabilities ε_i. Each of the probabilities in the better half of the $2m$ codewords must be less than ε since otherwise the sum over the other half would be at least $\frac{2m}{2}\varepsilon$ which contradicts $\sum_{i=1}^{2m} \varepsilon_i < m\varepsilon$. Therefore, throwing away the worse half of the codewords modifies (c^*, d^*) into into a (m, n)-channel code with rate at least $\rho(c^*, d^*) = C - \frac{2\varepsilon}{3} - \frac{1}{n}$ and $\varepsilon_{max} < \varepsilon$, as required. Taking n sufficiently large, and ε sufficiently small, we can approach any desired rate $R \leq C$.

(*Step 2: Rate is optimal*) Fix $\varepsilon > 0$ and assume for sufficiently large n there exists a (m, n) channel code with

$$\frac{\log(m)}{n} > R - \varepsilon \text{ and } \varepsilon_{max} < \varepsilon. \tag{4.4.1}$$

Let W be a random variable that is uniformly distributed on the messages $\{1, \cdots, m\}$ and as above, denote by $\hat{W}$ the decoded message. Then

$$\begin{aligned}
\log(m) &= H(W) \\
&= H(W|\hat{W}) + I(W;\hat{W}) \\
&\leq H(W|\hat{W}) + I(\mathbf{X};\mathbf{Y}) && \text{(by data processing inequality)} \\
&\leq H(W|\hat{W}) + \sum_{i=1}^{n} I(X_i;Y_i) && \text{(by information chain rule)} \\
&\leq H(W|\hat{W}) + nC && \text{(definition of } C\text{)} \\
&< 1 + \bar{\varepsilon}\log(m) + nC. && \text{(by Fano's inequality)}
\end{aligned}$$

Using $\bar{\varepsilon} \leq \varepsilon_{max} < \varepsilon$ and rearranging the above inequality gives

$$\frac{\log(m)}{n} < \frac{C + 1/n}{1 - \varepsilon}. \tag{4.4.2}$$

Using our assumption (4.4.1), this implies $R - \varepsilon < \frac{C+1/n}{1-\varepsilon}$. Sending $n \to +\infty$ and $\varepsilon \to 0$ we conclude $R \leq C$. □

Remark 4.19. The above proof even gives an asymptotic bound on the average error $\bar{\varepsilon}$ for a code (c, d) with rate $\rho(c, d) > C$. Rearranging the estimate (4.4.2) implies

$$\bar{\varepsilon} \geq 1 - \frac{1 + nC}{\log(m)} = 1 - \frac{C + 1/n}{\frac{1}{n}\log(m)}. \tag{4.4.3}$$

For large n, the right hand side is well approximated by $1 - \frac{C}{\frac{1}{n}\log(m)} = 1 - \frac{C}{\rho(c,d)}$.

Remark 4.20. The bound (4.4.3) implies a strictly positive average error for n big enough if the rate is bigger than C. To see this, assume by contradiction that the average error equals 0 for some n_0. Then we could transform this into a new (m^k, kn_0)-channel code by concatenating k codewords. But this channel has the same rate. Hence choosing k large enough contradicts the estimate (4.4.3). This is often called the weak converse of the channel coding theorem. There also exists a strong converse (which we do not prove) which shows that $\bar{\varepsilon} \to 1$ as $n \to +\infty$ if $\frac{\log(m)}{n} \geq C + \varepsilon$ for some $\varepsilon > 0$.

4.5 Channel codes

How to find a good channel code?

- If n is fixed we could try to search all possible codebooks. There are $|\mathcal{X}|^n$ codewords and approximately $|\mathcal{X}|^{mn}$ injective codes. If the rate of the code is assumed to be close to C then m is approximately $|\mathcal{X}|^{nC}$, hence we need to search over approximately $|\mathcal{X}|^{n|\mathcal{X}|^{nC}}$, which is computationally infeasible.

- We could try to use a randomly generated channel code as in the above proof. The above argument shows that is likely to be a good channel code for large n. Unfortunately, such a code is difficult to use in practice:
  - there are 2^{nR+1} codewords, i.e. to encode a message we need to store a table that grows exponentially with n;
  - the decoder needs to decide which of the 2^{nR+1} messages was transmitted, which again takes an exponential amount of time.

In fact, it took a long time after Shannon's proof of the existence of codes achieving rate C to find useful constructions. Breakthroughs are '72 Justesen, '93 Berrou et al, and '97 MacKay and Neal. The unifying idea of all these codes is to introduce some redundancy such that a perturbed message can still be recovered. There are two big classes of codes used commonly:

(1) block codes: to encode a block of information into a codeword but there is no dependence on past information. Examples include Hamming codes, Reed–Muller/Solomon codes, BCH codes, etc;

(2) convolutional codes: they are more complicated since they use dependency on the past inputs.

The search for optimal and practical codes is still an active area of research. In general this is a complicated topic that requires lots of algebra.

4.6 Block linear codes

Hamming created a family of error correcting codes, a particular example of block-linear codes, which perform well in many situations. Better codes exist, but their construction is typically very complicated.

The key idea of a linear error correcting code is to think of codewords in $|\mathcal{X}|^n$ as points in a space. Assuming $|\mathcal{X}| = 2$, the space we work in is the n-dimensional vector space over the finite field $\mathbb{F}_2$ (a.k.a. $\mathbb{Z}_2$, that is, $\{0, 1\}$ with addition and multiplication modulo 2). Our codewords can be directly interpreted as points in this space – for example $(1, 1, 0)$ is a vector in $\mathbb{F}_2^3$ corresponding to the codeword 110. These can also be thought of as the vertices of the n-dimensional unit hypercube. In order to give our space a metric, we have the following definition:

Definition 4.21. *(Hamming distance) For two length n vectors x, y, the Hamming distance between x and y is the number of entries in which x and y differ. In $\mathbb{F}_2^n$, we can see that this is equal to $\sum_i |x_i - y_i| = \sum_i |x_i - y_i|^2$, so the Hamming distance (formally) agrees with the ℓ_1/Manhattan distance and the Euclidean distance (but we should be careful, as we're not working over $\mathbb{R}$ or $\mathbb{C}$!). We call the corresponding norm the Hamming weight of the vector.*

If we can position our codewords in space such that every codeword is distance d from another codeword, then less than $\lfloor d/2 \rfloor$ errors in transmission will not prevent us decoding our message – we simply chose the closest codeword.

Suppose that for some $k < n$ we have at most 2^k codewords, which we can write as vectors $s \in \mathbb{F}_2^k$ (e.g. by ordering them and taking their dyadic expansions as codewords, or by applying a Tunstall code).

We will build new codewords w in $\mathbb{F}_2^n$ by the linear multiplication $w = sG$, where G is the 'generator matrix' of the code (a $k \times n$ matrix with values in $\mathbb{F}_2$). This places all our codewords on a k dimensional subspace of the n dimensional space (but as our field is finite, intuition is not always our friend here). We say G is of standard form if $G = [I_k|P]$ for some $k \times (n-k)$ dimensional matrix P. We say the space $\mathcal{C} = \{w = sG; s \in \mathbb{F}_2^k\}$ is the space of codewords. Given this construction, a generator matrix can be modified using the standard Gaussian elimination operations without changing the space of codewords.

Warning: You need to remember that all our calculations are in $\mathbb{F}_2^n$, in particular $1 + 1 = 0$!

Definition 4.22. *We say an error correcting code satisfying the description above is a $[n, k, d]_2$ code.*

Proposition 4.23. *(Singleton–Joshi–Komamiya bound; Komamiya (1953), Joshi (1958)) For any $[n, k, d]_2$ code, we have $d \leq n - k + 1$.*

Proof. By construction, there are at most 2^k codewords in our space. However, given the minimal distance between codewords is d, we can remove the first $d-1$ entries of each codeword, and still have all codewords distinct. The resulting codes are of length $n - d + 1$, from which it follows that $2^k \leq 2^{n-d+1}$. Rearrangment gives the result. □

We say a $(n-k) \times n$ matrix H is a (parity) check matrix for G if the kernel of H is $\mathcal{C}$. In other words, $wH^\top = 0$ for all codewords w, or equivalently $GH^\top$ is a zero matrix. If G is in standard form, the matrix $H = [P^\top|I_{n-k}]$ is a parity check matrix for G. By the rank-nullity theorem, if H has rank $n-k$, we can easily check that a vector in $\mathbb{F}_2^n$ is a codeword if and only if $wH^\top = 0$.

Example 4.24. Take $n = 3, k = 2$, with $G = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$ and $H = [1, 1, 1]$. Then we see that every codeword satisfies $wH^\top = w_1 + w_2 + w_3 = 0$. The final element of the codeword has the interpretation of being a check of the parity of the sum of the previous elements.

By linearity, we can now compute the minimal distance d between codewords:

$$d = \min_{w \neq w' \in \mathcal{C}} |w - w'| = \min_{s \neq s' \in \mathbb{F}_2^k} |sG - s'G| = \min_{\hat{s} = s - s' \neq 0 \in \mathbb{F}_2^k} |\hat{s}G| = \min_{\hat{w} \neq 0 \in \mathcal{C}} |\hat{w}|.$$

In other words, the minimal distance is the same as the minimal weight of all codewords.

Proposition 4.25. *The minimal distance between codewords is equal to the minimum number of linearly dependent columns in a check matrix H.*

Proof. We know that $Hw^\top = 0$ for all codewords. Writing $Hw^\top = \sum_i w_i h_i$, for h_i the columns of H, we see that the minimal codeword weight is at least the number of linearly dependent columns in H.

Conversely, take a check matrix, and write $\{h_i\}$ for the columns of H. If a subset of m columns is linearly dependent, there exists $c \in \mathbb{F}_2^n$ such that $0 = \sum_i c_i h_i$, and $|(c_1, ..., c_n)| = m$. This c is then a codeword with weight m. Hence the minimal codeword weight is at most the number of linearly dependent columns in H. □

There are various constructions of these codes. Hamming gives one possible construction:

Definition 4.26. *For $n = 2^m - 1$, $k = n - m$ and $d = 3$, let H be the matrix with columns given by all pairwise linearly independent vectors of length m. This gives the 'Hamming code'.*

Example 4.27. For $m = 3$, we get Hamming's $[7, 4, 3]_2$ code (which can correct a single error), with

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}, \qquad H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

The Hamming code is used extensively, for example it is the basis of error correction within many RAM circuits.

Other related codes can be built by modifying the Hamming construction slightly:

Example 4.28. Adding an additional parity bit (a row of 1's to H) defines the *extended Hamming code*. This is a SECDED code - a single error correcting and double error detecting (it can detect but not correct two errors) code. A corresponding G can be found by solving $GH^\top = 0$. The code words all have weight 4 and this is a Hamming $[8, 4, 4]$ code. By applying row operations to H and G, these can be shown to have the equivalent standard form $G = [I_4|1 - I_4]$ and $H = [1 - I_4|I_4]$, where 1 denotes a matrix of 1s.

Example 4.29. If H is the parity check matrix of a code, we obtain the *dual code* by taking the (non-standard) generator $\tilde{G} = H$. Taking the dual of the extended Hamming code defines the (augmented) *Walsh–Hadamard code*, which is a $[2^k, k + 1, 2^{k-1}]_2$ code. Alternative constructions, based on Hadamard matrices, are typically preferred. These codes are used in CDMA (code-division multi access) standards, such as 3G, to allow multiple users to communicate over the same channel, as the codewords are orthogonal, so each user can send their codeword without interfering with others.

Example 4.30. Hsiao (1970) gives a variation on the Hamming construction, which forces the check matrix H to have only odd numbers of entries in each column. For larger codes, this can be shown to improve computational efficiency.

Example 4.31. Polar codes, and the closely related Reed–Muller codes (which are $[2^m, \sum_{i \leq r} \binom{m}{i}, 2^{m-r}]_2$ codes constructed using polynomials in finite fields), are the basis for error correction in the 5G mobile standard.

Chapter 5

Noisy Channels with non-iid input

It is natural to ask whether one can combine Shannon's two theorems: given a signal such as digitised speech, the obvious approach is to first apply symbol coding (Theorem 3.4) for compression, and then apply channel coding (Theorem 4.15) to send this compressed signal through our channel. Two questions arise: Firstly, is this two-stage approach optimal, or would it be better to directly feed the digitized signal into channel coding without an initial compression layer? Secondly, as the channel input will not be an i.i.d. sequence, can we extend our results to cover this case?

5.1 Channel coding with non-iid input

We first address the second question by showing that the notion of entropy extends to sequences of (possibly dependent) random variables. We use this in the next section to answer the first question of optimality.

Definition 5.1. *A discrete time stochastic process with state space $\mathcal{X}$ is a sequence $X = (X_i)_{i\geq 1}$ of $\mathcal{X}$-valued random variables. We say that a stochastic process is stationary if*

$$\mathbb{P}(X_1 = x_1, \cdots, X_n = x_n) = \mathbb{P}(X_{1+j} = x_1, \cdots, X_{n+j} = x_n)$$

for all integers n, j and $x_1, \cdots, x_n \in \mathcal{X}$.

A special case is a stochastic process with X_i i.i.d., but much more complicated statistical dependencies can occur between the X_i.

Definition 5.2. *The entropy rate of a stochastic process $X = (X_i)_i$ is defined as*

$$\mathcal{H}(X) = \lim_{n\to+\infty} \frac{1}{n} H(X_1, \cdots, X_n),$$

whenever this limit exists.

Obviously, if X_i are i.i.d., then the entropy rate exists and $\mathcal{H}(X) = \lim_{n\to+\infty} \frac{1}{n}(H(X_1) + \cdots + H(X_n)) = H(X_1)$. However, for the case when X_i are independent but not identically distributed the above limit

does not necessarily exist. For example the binary variables X_i with

$$\mathbb{P}(X_i = 1) = 0.5 \text{ for } \log(\log(i)) \in (2k, 2k+1] \quad \text{and } \mathbb{P}(X_i = 1) = 0 \text{ for } \log(\log(i)) \in (2k+1, 2k+2]$$

where k can be any number in $\{0, 1, 2, \cdots\}$. This construction gives long stretches with $H(X_i) = 1$ followed by exponentially longer stretches of $H(X_i) = 0$, hence the running average will oscillate between 0 and 1.

Theorem 5.3. *For a stationary stochastic processes X, the entropy rate exists and*

$$\mathcal{H}(X) = \lim_{n \to +\infty} H(X_n | X_{n-1}, \cdots, X_1).$$

We prepare the proof with two Lemmas.

Lemma 5.4. *For a stationary stochastic process X, $n \mapsto H(X_n|X_{n-1}, \cdots, X_1)$ is non-increasing and $\lim_{n\to+\infty} H(X_n|X_{n-1}, \cdots, X_1)$ exists.*

Proof. For any integer n,

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

where we used that conditioning reduces entropy for the inequality, and the equality is due to the stationarity of the process X. Since $H(X_n|X_{n-1}, \cdots, X_1) \geq 0$, the limit exists. □

Lemma 5.5. *(Cesàro mean). If $\lim_{n\to+\infty} a_n = a$, then $\lim_{n\to+\infty} \frac{1}{n} \sum_{i=1}^{n} a_i = a$.*

Proof. For any $\varepsilon > 0$, there exists a n_0 such that for all $n \geq n_0$, , $|a_n - a| < \varepsilon$. Hence

$$\begin{aligned} \left| \frac{1}{n} \sum_{i=1}^{n} a_i - a \right| &\leq \frac{1}{n} \sum_{i=1}^{n} |a_i - a| \\ &\leq \frac{1}{n} \sum_{i=1}^{n_0} |a_i - a| + \frac{n - n_0}{n} \varepsilon. \end{aligned}$$

Sending $n \to +\infty$ makes the first term vanish and the result follows. □

We can now give a proof of Theorem 5.3.

Proof of Theorem 5.3. By the chain rule for conditional entropy,

$$\frac{H(X_1, \cdots, X_n)}{n} = \frac{1}{n} \sum_{i=1}^{n} H(X_i | X_{i-1}, \cdots, X_1).$$

By Lemma 5.4 the conditional entropies converge. Using Cesàro means as in Lemma 5.5, the above running average of conditional entropies converges to $\lim_{n\to+\infty} H(X_n|X_{n-1}, \cdots, X_1)$. □

A general result in this direction is the Shannon–McMillan–Breiman Theorem[1]:

[1] The version here is due to Breiman, there are many extensions (to a.s. convergence, non-stationary sequences, etc); see [1] for reference

Theorem 5.6. *(Shannon–McMillan–Breiman). Let $X_1, X_2, \cdots$ be an ergodic and stationary sequence of random variables in a finite state space $\mathcal{X}$. Then*

$$-\frac{1}{n}\log(p_{X_1,\cdots,X_n}(X_1,\cdots,X_n)) \overset{\text{in prob.}}{\longrightarrow} \mathcal{H}, \text{ as } n \to +\infty,$$

(Loosely speaking, a sequence is ergodic if the time average over one realisation converges to the expectation. The class of of stationary and ergodic processes is large and covers many important processes.) To prove this result, one can modify Theorem 2.12 and adapt Shannon's block coding argument of Theorem 3.4.

We now turn to Markov chains, a class of stochastic processes.

Definition 5.7. *A discrete stochastic process $X = (X_i)_{i\geq 1}$ is a Markov chain on a state space $\mathcal{X}$ if*

$$\mathbb{P}(X_n = x_n \,|\, X_{n-} = x_{n-1}, \cdots, X_1 = x_1) = \mathbb{P}(X_n = x_n \,|\, X_{n-1} = x_{n-1})$$

for all n and all $x_1, \cdots, x_n \in \mathcal{X}$.

A Markov chain is time-invariant (or time-homogenous) if

$$\mathbb{P}(X_{n+1} = b | X_n = a) = \mathbb{P}(X_2 = b \,|\, X_1 = a)$$

for all n and all $a, b \in \mathcal{X}$.

A time-invariant Markov chain with state space $\mathcal{X} = \{x_1, \cdots, x_m\}$ is characterised by its initial state X_1 and its probability transition matrix

$$P = (P_{i,j})_{m\times m} \text{ where } P_{i,j} := \mathbb{P}(X_2 = x_j \,|\, X_1 = x_i).$$

In this case, the pmf of X_{n+1} is given as $p_{X_{n+1}}(x_j) = \sum_i p_{X_n}(x_i)P_{i,j}$.

Given a time-invariant Markov process X, the distribution p_{X_n} on $\mathcal{X}$ is called a stationary distribution of X if $p_{X_{n+1}} = p_{X_n}$. Hence, a pmf μ on $\mathcal{X}$ is a stationary distribution, if $\mu_j = \sum_i \mu_i P_{i,j}$ for all j, where $\mu_i = \mu(x_i)$, or in matrix notation (with $\mu = (\mu_1, \cdots, \mu_m)$)

$$\mu = \mu P.$$

Remark 5.8. If a time-invariant Markov chain is irreducible (i.e. for any $x_1, x \in \mathcal{X}$ there exists $t > 0$ such that $\mathbb{P}(X_t = x | X_1 = x_1) > 0$) then it has a unique stationary distribution; if the Markov chain is also aperiodic (i.e. the irreducibility condition holds for all t sufficiently large, or equivalently for two coprime values of t), then for any $x_0 \in \mathcal{X}$ we have the geometric convergence $|\mu - x_0 P^n| \leq c(1-\varepsilon)^n$ for some $c, \varepsilon > 0$ (this is a consequence of the Perron–Frobenius theorem). This result is known as the geometric ergodicity of the Markov chain. We will not prove this result here.

A time-invariant Markov chain with stationary distribution μ and initial state $X_1 \sim \mu$ is a stationary stochastic process and its entropy rate is given by

$$\mathcal{H}(X) = \lim_{n\to+\infty} H(X_n \,|\, X_{n-1}, \cdots, X_1) = \lim H(X_n \,|\, X_{n-1}) = H(X_2|X_1).$$

Using the definition of conditional entropy this becomes

$$\mathcal{H}(X) = \sum_i \mathbb{P}(X_1 = x_i) H(X_2|X_1 = x_i) = -\sum_i \mu_i \left(\sum_j P_{i,j} \log(P_{i,j}) \right) = -\sum_{i,j} \mu_i P_{i,j} \log(P_{i,j}).$$

While we will focus on stationary processes (so $X_1 \sim \mu$), the geometric ergodicity of a Markov chain can be used to show that the entropy rate is well defined for any starting distribution, and equals the rate for the stationary case.

Example 5.9. Let $X = (X_i)$ be Markov chain with two states $\mathcal{X} = \{a, b\}$ and $\mathbb{P}(X_2 = b|X_1 = a) = \alpha, \mathbb{P}(X_2 = a|X_1 = b) = \beta$, that is

$$P = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix}.$$

Then the stationary distribution is $\mu(a) = \frac{\beta}{\alpha+\beta}, \mu(b) = 1 - \mu(\alpha)$. If $X_1 \sim \mu$, then

$$\mathcal{H}(X) = \frac{\beta}{\alpha+\beta} H(\alpha) + \frac{\alpha}{\alpha+\beta} H(\beta),$$

where $H(\alpha)$ denotes the entropy of a Bernoulli random variable with probability α (and similarly $H(\beta)$).

Example 5.10. Consider a connected[2] graph (V, E) with vertices $V = \{1, \cdots, m\}$ and without self-connection. Associate with the edge connecting vertices i and j a weight $w_{i,j} = w_{j,i} \geq 0$ (if there's no edge, set $w_{i,j} = 0$). Define a Markov chain on the set of vertices V by

$$P_{i,j} = \mathbb{P}(X_{n+1} = j \,|\, X_n = i) = \frac{w_{i,j}}{\sum_{k=1}^{m} w_{i,k}}.$$

(Choose the next vertex at random from the neighbouring vertices, with probabilities proportional to the weight of the connecting edge). We can guess the stationary distribution: the probability of being at vertex i should be proportional to the total weight of the edges emanating from this vertex. That is, if we denote the total weight of edges connecting to vertex i by $w_i = \sum_j w_{j,i}$, and the sum of weight of all edges by $w = \sum_{j>i} w_{i,j}$. Then $\sum_i w_i = 2w$ and we expect that $\mu_i = \frac{w_i}{2w}$. Indeed, we can directly verify $\mu P = \mu$:

$$\sum_i \mu_i P_{i,j} = \sum_i \frac{w_i}{2w} \frac{w_{i,j}}{w_i} = \frac{1}{2} \sum_i \frac{w_{i,j}}{2} = \frac{w_j}{2w} = \mu_j.$$

It is interesting to note that μ_i does not change if the edge weights connecting to vertex i stay the same, but the other weights are changed subject to having the same total weight. To calculate the entropy rate

$$\begin{aligned}
\mathcal{H}(X) &= H(X_2|X_1) = -\sum_i \mu_i \sum_j P_{i,j} \log(P_{i,j}) \\
&= -\sum_i \frac{w_i}{2w} \sum_j \frac{w_{i,j}}{w_i} \log\left(\frac{w_{i,j}}{w_i}\right) \\
&= -\sum_{i,j} \frac{w_{i,j}}{2w} \log\left(\frac{w_{i,j}}{w_i}\right) \\
&= -\sum_{i,j} \frac{w_{i,j}}{2w} \log\left(\frac{w_{i,j}}{w_i} \frac{2w}{2w}\right) \\
&= -\sum_{i,j} \frac{w_{i,j}}{2w} \log\left(\frac{w_{i,j}}{2w}\right) + \sum_{i,j} \frac{w_{i,j}}{2w} \log\left(\frac{w_i}{2w}\right) \\
&= -\sum_{i,j} \frac{w_{i,j}}{2w} \log\left(\frac{w_{i,j}}{2w}\right) + \sum_{i} \frac{w_i}{2w} \log\left(\frac{w_i}{2w}\right)
\end{aligned}$$

[2] A graph is connected if every pair of vertices can be connected by a path of edges.

This final quantity can be expressed in terms of the entropies of the distribution (on i,j) with probabilities $\frac{w_{i,j}}{2w}$, and the distribution (on i) with probabilities $\frac{w_i}{2w}$

Example 5.11. Consider English-language text, with alphabet $|\mathcal{A}| = 27$. We can define a Markov chain model for the sequence of letters by estimating $\{\mathbb{P}(L_t = a|L_{t-1} = b)\}_{a,b\in\mathcal{A}}$ from data, where L_t is the tth observed letter. More generally, we can take $X_{t-1} = [L_{t-1}, L_{t-2}, ..., L_{t-k}]$ for a model with $\mathcal{X} = \mathcal{A}^k$ where the next letter depends on the preceeding k letters, and estimate $\{\mathbb{P}(L_t = a|L_{t-1} = b_1, ..., L_{t-k} = b_k)\}_{a,b_1,...,b_k\in\mathcal{A}}$. In practice, $k \geq 5$ is needed to reasonably resemble English text.

A similar setting is used by large language models (e.g. ChatGPT, Bard, Sydney), where the previous k words are used, and a predictive model is learned for the next word. As there are a large number of possible words, this requires a very large model, but is conceptually in the same class.

In practice, one is often not directly interested in the Markov chain $X = (X_i)$ but in understanding a process Y defined by a function of X, i.e., $Y_i = \phi(X_i)$. For example, think of X as a complicated system that evolves over time but we only observe the current state of the system partially. A basic question is to determine the entropy rate of the stochastic process Y. This is a complicated question since in general Y itself is not a Markov chain so we can't directly apply the results of the previous section. However, we know that $\mathcal{H}(Y)$ is well-defined since Y is stationary.

A first approach is to simply estimate $\mathcal{H}(Y)$ by the first n observations as $H(Y_n \,|\, Y_{n-1}, \cdots, Y_1)$. However, the convergence $\mathcal{H}(Y) = \lim_n H(Y_n|Y_{n-1}, \cdots, Y_1)$ can be very slow so we have no means to decide whether this estimate is good for a given n! The theorem below shows that the difference $H(Y_n \,|\, Y_{n-1}, \cdots, Y_1) - H(Y_n|Y_{n-1}, \cdots, Y_1, X_1)$ gives guarantees for this estimate.

Theorem 5.12. *Let $X = (X_i)_{i\in\mathbb{Z}}$ be a stationary Markov chain and $\phi : \mathcal{X} \longrightarrow \mathcal{Y}$. Let $Y = (Y_i)_{i\in\mathbb{Z}}$ with $Y_i := \phi(X_i)$. Then*

$$H(Y_n|Y_{n-1}, \cdots, Y_1, X_1) \leq \mathcal{H}(Y) \leq H(Y_n|Y_{n-1}, \cdots, Y_1)$$

and $\mathcal{H}(Y) = \lim_{n\to+\infty} H(Y_n|Y_{n-1}, \cdots, Y_1, X_1) = \lim_{n\to+\infty} H(Y_n|Y_{n-1}, \cdots, Y_1)$.

Since $H(Y_n|Y_{n-1}, \cdots, Y_1)$ converges monotonically from above to $\mathcal{H}(Y)$, the theorem follows by combining the following two lemmas.

Lemma 5.13. $H(Y_n|Y_{n-1}, \cdots, Y_2, X_1) \leq \mathcal{H}(Y)$.

Proof. Using that $Y_1 = \phi(X_1)$, the Markovianity of X, that $Y_i = \phi(X_i)$ we get for any positive integer k that

$$\begin{aligned}
H(Y_n|Y_{n-1}, \cdots, Y_2, X_1) &= H(Y_n|Y_{n-1}, \cdots, Y_2, Y_1, X_1) \\
&= H(Y_n|Y_{n-1}, \cdots, Y_2, Y_1, X_1, X_0, X_{-1}, \cdots, X_{-k}) \\
&= H(Y_n|Y_{n-1}, \cdots, Y_2, Y_1, X_1, X_0, X_{-1}, \cdots, X_{-k}, Y_0, \cdots, Y_{-k}) \\
&\leq H(Y_n|Y_{n-1}, \cdots, Y_1, Y_0, \cdots, Y_{-k}) \\
&= H(Y_{n+k+1}|Y_{n+k}, \cdots, Y_1),
\end{aligned}$$

where the inequality is because the conditioning reduces entropy. So, as this holds for each integer k we have

$$H(Y_n|Y_n, ..., Y_2, X_1) \leq \lim_k H(Y_{n+k+1}|Y_{n+k}, \cdots, Y_1) = \mathcal{H}(Y).$$

□

Lemma 5.14. *$H(Y_n|Y_{n-1},\cdots,Y_1) - H(Y_n|Y_{n-1},\cdots,Y_1,X_1) \to 0$ as $n \to +\infty$.*

Proof. $I(X_1;Y_n\,|\,Y_{n-1},\cdots,Y_1) = H(Y_n\,|\,Y_{n-1},\cdots,Y_1) - H(Y_n\,|\,Y_{n-1},\cdots,Y_1,X_1)$.
Since $I(X_1;Y_n,Y_{n-1},\cdots,Y_1) \leq H(X_1)$ and $n \mapsto I(X_1;Y_n,Y_{n-1},\cdots,Y_1)$ is increasing, the limit

$$\lim_n I(X_1;Y_n,Y_{n-1},\cdots,Y_1) \leq H(X_1)$$

exists. By the chain rule,

$$I(X_1;Y_n,Y_{n-1},\cdots,Y_1) = \sum_{i=1}^{n} I(X_1;Y_i\,|\,Y_{i-1},\cdots,Y_1),$$

so combining with the above we get

$$\sum_{i=1}^{\infty} I(X_1;Y_i\,|\,Y_{i-1},\cdots,Y_1) \leq H(X_1) < \infty,$$

thus $\lim_{n\to+\infty} I(X_1;Y_n\,|\,Y_{n-1},\cdots,Y_1) = 0$. □

In the case where the function ϕ is a random function of X we are in the setting of channel coding.

5.2 Combining symbol and channel coding for DMCs

We now have a useful definition for the entropy of a non-i.i.d. source process. We will use this to understand the interaction between symbol and channel coding.

Consider a source that generates symbols from a finite set $\mathcal{V}$. We model this source as a discrete stochastic process $V = (V_i)$ with state space $\mathcal{V}$. Our goal is to transmit a sequence of symbols $V^n := (V_1,\cdots,V_n)$ over a DMC. Therefore we use an encoder $c : \mathcal{V}^n \longrightarrow \mathcal{X}^n$ and recover V^n from the output sequence Y^n by using a decoder $d : \mathcal{Y}^n \longrightarrow \mathcal{V}^n$. We want to do this in such a way that $\mathbb{P}(V^n \neq \hat{V}^n)$ is small.

Theorem 5.15. *Let $(\mathcal{X}, M, \mathcal{Y})$ be a DMC with channel capacity C. Let $V = (V_i)_{i\geq 1}$ be a discrete stochastic process in a finite state space $\mathcal{V}$. If V satisfies the AEP and*

$$\mathcal{H}(V) < C,$$

then for every $\varepsilon > 0$ there exists an $n \geq 1$, a map $c : \mathcal{V}^n \longrightarrow \mathcal{X}^n$, and a map $d : \mathcal{Y}^n \longrightarrow \mathcal{V}$ such that $\mathbb{P}(V^n \neq \hat{V}^n) < \varepsilon$. Conversely, for any stationary stochastic process V, if $\mathcal{H}(V) > C$, there exists a constant $\delta > 0$ such that $\mathbb{P}(V^n \neq \hat{V}^n) > \delta$ for any coder-decoder pair, for any $n \geq 1$.

Sketch of Proof. As V satisfies the AEP, there exists a typical set $\mathcal{T}_\varepsilon^{(n)}$ of size $|\mathcal{T}_\varepsilon^{(n)}| \leq 2^{n(\mathcal{H}(V)+\varepsilon)}$ such that $\mathbb{P}(V^n \in \mathcal{T}_\varepsilon^{(n)}) \geq 1-\varepsilon$. Now consider a coder that only encodes elements in $\mathcal{T}_\varepsilon^{(n)}$ and elements in $\mathcal{V}^n \backslash \mathcal{T}_\varepsilon^{(n)}$ are all encoded randomly using the rest of the codewords not used for those in $\mathcal{T}_\varepsilon^{(n)}$. We need at most

$$n(\mathcal{H}(V)+\varepsilon)$$

bits to index elements in $\mathcal{T}_\varepsilon^{(n)}$. Using channel coding we can transmit such an index with probability of error less than ε given the fact

$$\mathcal{H}(V) + \varepsilon = R < C.$$

The decoder reconstructs V^n by enumerating the typical set $\mathcal{T}_\varepsilon^{(n)}$ and decoding the received index $Y^n = (Y_1, \cdots, Y_n)$ to get $\hat{V}^n$. Then for a large enough n,

$$\mathbb{P}(V^n \neq \hat{V}^n) \leq \mathbb{P}(V^n \notin \mathcal{T}_\varepsilon^{(n)}) + \mathbb{P}(d(Y^n) \neq V^n \,|\, V^n \in \mathcal{T}_\varepsilon^{(n)}) \leq \varepsilon + \varepsilon.$$

This shows the first part of the theorem (achievability). For the second part (optimality) we need to show that

$$\mathbb{P}(V^n \neq \hat{V}^n) \to 0$$

implies $\mathcal{H}(V) \leq C$ for any sequence (c^n, d^n) of channel codes. By Fano's inequality,

$$\begin{aligned} H(V^n|\hat{V}^n) &\leq 1 + \mathbb{P}(\hat{V}^n \neq V) \log(|\mathcal{V}^n|) \\ &= 1 + \mathbb{P}(\hat{V}^n \neq V) n \log(|\mathcal{V}|). \end{aligned}$$

Now

$$\begin{aligned} \mathcal{H}(V) &\leq \frac{H(V_1, \cdots, V_n)}{n} \\ &= \frac{1}{n} H(V_1, \cdots, V_n \,|\, \hat{V}_1, \cdots, \hat{V}_n) + \frac{1}{n} I(V^n; \hat{V}^n) \\ &\leq \frac{1}{n}\left[1 + \mathbb{P}(V^n \neq \hat{V}^n) n \log(|\mathcal{V}|)\right] + \frac{1}{n} I(V^n; \hat{V}^n) \\ &\leq \frac{1}{n}\left[1 + \mathbb{P}(V^n \neq \hat{V}^n) n \log(|\mathcal{V}|)\right] + \frac{1}{n} I(X_1, \cdots, X_n; Y_1, \cdots, Y_n) \\ &\leq \frac{1}{n} + \mathbb{P}(V^n \neq \hat{V}^n) \log(|\mathcal{V}|) + C, \end{aligned}$$

where we used: the definition of entropy rate, the definition of mutual information, Fano's inequality, the data processing inequality, and finally, the definition of capacity of a DMC. Letting $n \to +\infty$ finishes the proof since

$$\mathcal{H}(V) \leq \log(|\mathcal{V}|) \lim_{n \to +\infty} \mathbb{P}(V^n \neq \hat{V}^n) + C = C.$$

□

We emphasise that above theorem makes no assumptions on the stochastic process V other than that the AEP holds; the sequence of random variables $(V_1, \cdots, V_n)$ can have very complicated dependencies. Most importantly, the theorem implies that a two-stage approach – given by firstly using symbol coding and then applying channel coding – achieves the same rates as applying source coding directly. This two-stage approach is advantageous from an engineering perspective since it divides a complicated problem into two smaller problems. On the other hand, it still only gives us an existence result – actually finding good codes which achieve this bound, for a typical problem, remains difficult!

To sum up: source coding compresses the information using that, by the AEP, there exists a set of small cardinality $\approx 2^{nH}$ that carries most of the probability mass. Hence, we can use H bits per symbol in a symbol code to compress the source. Channel coding uses that by the joint AEP, we have for large n with high probability that input and output are jointly typical; only with probability $\approx 2^{-nI}$ any other codeword will be jointly typical. Thus we can use 2^{nI} codewords. Theorem 5.15 shows that we can design the source code and channel code separately without loss of performance.

5.3 Decoding from a noisy non-iid channel

Given we have received a message which arose from a Markov chain source, and was encoded and transmitted over a noisy discrete memoryless channel, the final major task is to reconstruct the original signal, in a reasonably efficient manner. This is a special case of a more general problem in time series analysis – reconstructing the state of hidden Markov processes from observations.

We will do this in two stages. We will first consider an algorithm due to Wonham which efficiently calculates the posterior probabilites of the source messages at each time, given the observations up to that time. We will then consider an extension of this due to Viterbi, which calculates the most-likely source message from the entire sequence of observations.

5.3.1 Wonham filter

For simplicity, we will assume our Markov chain is initialized according to some distribution $X_0 \sim \mu_0$ (which may or may not be stationary).

We begin by associating the states of our Markov chain with the basis row-vectors in $\mathbb{R}^{|\mathcal{X}|}$, which makes the algebra easier. That is, if $|\mathcal{X}| = 3$, we have states $[1, 0, 0]$, $[0, 1, 0]$ and $[0, 0, 1]$. With this notation, we see that xP is the probability vector describing the distribution of states of $X_2|(X_1 = x)$. Furthermore, as our states are written in this way, we know that

$$\mathbb{P}[X_2 = x|X_1 = \tilde{x}] = \tilde{x}Px^\top.$$

Recall that we have defined the emission matrix M such that xM is the vector of probabilities of each observed value, given the current state is x. If we identify $\mathcal{Y}$ with the basis column-vectors in $\mathbb{R}^{|\mathcal{Y}|}$, we have

$$\mathbb{P}(Y_t = y|X_t = x) = xMy.$$

In particular, we can compute

$$\mathbb{P}(X_t = x, Y_t = y|X_{t-1} = \tilde{x}) = \mathbb{P}(Y_t = y|X_t = x)\mathbb{P}(X_t = x|X_{t-1} = \tilde{x}) = (xMy)(\tilde{x}Px^\top).$$

The definition of conditional probability then lets us calculate

$$\mathbb{P}(X_t = x|Y_t = y, X_{t-1} = \tilde{x}, X_{s<t-1}) = \frac{\mathbb{P}(X_t = x, Y_t = y|X_{t-1} = \tilde{x})}{\sum_{z\in\mathcal{X}} \mathbb{P}(X_t = z, Y_t = y|X_{t-1} = \tilde{x})} = \frac{(xMy)(\tilde{x}Px^\top)}{\sum_z (zMy)(\tilde{x}Pz^\top)}.$$

As $\sum_x \mathbb{P}(X_t = x|Y_t = y, X_{t-1} = \tilde{x}) = 1$, we can ignore the denominator on the right hand side, and obtain the probability up-to-renormalization, namely

$$\mathbb{P}(X_t = x|Y_t = y, X_{t-1} = \tilde{x}) \propto (xMy)(\tilde{x}Px^\top)$$

We now do some algebraic rearrangement:

$$(xMy)(\tilde{x}Px^\top) = \tilde{x}Px^\top xMy = \tilde{x}(P)(x^\top x)(My)$$

and observe that $x^\top x = \mathrm{diag}(x)$, and $\mathrm{diag}(x)My = \mathrm{diag}(My)x^\top$. Therefore,

$$\mathbb{P}(X_t = x|Y_t = y, X_{t-1} = \tilde{x}) \propto \tilde{x}(P\mathrm{diag}(My))x^\top.$$

In other words, $\tilde{x}(P\text{diag}(My))$ gives (up to renormalization), the vector of probabilities for each state of X_t, given the prior state $\tilde{x}$ and the observation y.

We can extend this further, by noticing that (for any event A)

$$\mathbb{P}(X_t = x|A) = \mathbb{E}[X_t|A]x^\top.$$

(as X takes values in basis vectors). Therefore, we can write our conditional probability vector as

$$\mathbb{E}[X_t|Y_t = y, X_{t-1} = \tilde{x}] \propto \tilde{x}(P\text{diag}(My)).$$

Using this, and the fact that X_t, Y_t are independent of X_{t-2} and Y_{t-2} given X_{t-1}, we get a simple recursion:

Proposition 5.16. *(The Wonham filter) For a Markov chain with $X_0 \sim \mu_0$, the conditional probability vector $\mu_t = \mathbb{E}[X_t|\{Y_s = y_s\}_{s\le t}]$ satisfies*

$$\mu_t \propto \mu_{t-1}P\text{diag}(My_t)$$

Proof. As described above, we know

$$\begin{aligned}
\mu_t &= \mathbb{E}[X_t|\{Y_s = y_s\}_{s\le t}] \\
&= \sum_{\tilde{x}} \mathbb{P}(X_{t-1} = \tilde{x}|\{Y_s = y_s\}_{s\le t-1})\mathbb{E}[X_t|X_{t-1} = \tilde{x}, Y_t = y_t, \{Y_s = y_s\}_{s\le t-1}] \\
&= \sum_{\tilde{x}} (\mu_{t-1}\tilde{x}^\top)\mathbb{E}[X_t|X_{t-1} = \tilde{x}, Y_t = y_t] \\
&\propto \sum_{\tilde{x}} (\mu_{t-1}\tilde{x}^\top)\tilde{x}(P\text{diag}(My_t)) \\
&= \mu_{t-1}\Big(\sum_{\tilde{x}} \text{diag}(\tilde{x})\Big)(P\text{diag}(My_t)) = \mu_{t-1}(P\text{diag}(My_t)).
\end{aligned}$$

□

This gives a straightforward algorithm to estimate X_t from observations of $\{Y_s\}_{s\le t}$, we simply apply the above recursion, and renormalize at each step (or whenever is needed for stability).

Example 5.17. Suppose

$$P = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} \text{ and } M = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix}.$$

Assume we know $X_0 = x_1$ and observe the sequence $Y = (1, 2, 2, 1)$. We compute

$$P\text{diag}(\text{Me}_1) = \begin{bmatrix} 0.63 & 0.06 \\ 0.36 & 0.12 \end{bmatrix}, \qquad P\text{diag}(\text{Me}_2) = \begin{bmatrix} 0.07 & 0.24 \\ 0.04 & 0.48 \end{bmatrix}$$

Our sequence of estimated states are then given by multiplying and renormalizing

Time	0	1	2	3	4
Unnormalized prob	1	0.63	0.046	0.01	0.042
	0	0.06	0.18	0.098	0.012
Normalized prob	1	0.91	0.21	0.10	0.77
	0	0.09	0.79	0.90	0.23

5.3.2 Viterbi algorithm

The Wonham filter gives us the best estimate of X_t at each time, based on the current observations. However, in the coding context we usually wish to determine the whole path of X, rather than simply its present value. This is the purpose of the Viterbi algorithm, which extends the Wonham filter to give the most likely path.

Based on our calculations above, at time t we know that

$$\mathbb{P}\Big(X_t = x_t, Y_t = y_t \Big| X_{t-1} = x_{t-1}, \{X_s\}_{s<t}, \{Y_s\}_{s<t}\Big) \propto x_{t-1} P \mathrm{diag}(My_t) x_t^\top.$$

Therefore, the probability of the joint path of X and Y can be written recursively as

$$\mathbb{P}\Big(\{X_s = x_s\}_{s\le t}, \{Y_s = y_s\}_{s\le t}\Big) \propto (x_{t-1} P \mathrm{diag}(My_t) x_t^\top) \mathbb{P}\Big(\{X_s = x_s\}_{s<t}, \{Y_s = y_s\}_{s<t}\Big).$$

where the missing normalization constant doesn't depend on $\{x_s\}_{s\le t}$. However, by Bayes' rule

$$\mathbb{P}\Big(\{X_s = x_s\}_{s\le t} \Big| \{Y_s = y_s\}_{s\le t}\Big) = \frac{\mathbb{P}\Big(\{X_s = x_s\}_{s\le t}, \{Y_s = y_s\}_{s\le t}\Big)}{\mathbb{P}\Big(\{Y_s = y_s\}_{s\le t}\Big)}$$

so

$$\mathbb{P}\Big(\{X_s = x_s\}_{s\le t} \Big| \{Y_s = y_s\}_{s\le t}\Big) \propto (x_{t-1} P \mathrm{diag}(My_t) x_t^\top) \mathbb{P}\Big(\{X_s = x_s\}_{s<t} \Big| \{Y_s\}_{s<t}\Big) \tag{5.3.1}$$

with a normalization constant independent of $\{x_s\}_{s\le t}$.

Proposition 5.18. *Consider paths for X ending in x. The highest posterior probability for such paths is*

$$\pi_t^x = \max_{\{x_s\}_{s<t}} \mathbb{P}\Big(\{X_s = x_s\}_{s<t}, X_t = x \Big| \{Y_s = y_s\}_{s\le t}\Big),$$

and this is achieved by a path denoted $\chi_t^x = \{x_0, ..., x_{t-1}, x\} \in \mathcal{X}^t$.

The quantities χ_t^x and π_t^x have initial values $\chi_0^x = \{x\}$ and $\pi_0^x = \mu_0 x^\top$, and satisfy the recursion

$$\begin{aligned} \chi_t^x &= \big[\chi_{t-1}^{\tilde{x}}, x\big], \\ \pi_t^x &\propto \big(\tilde{x} P \mathrm{diag}(My_t) x^\top\big) \pi_{t-1}^{\tilde{x}}, \\ \textit{where} \qquad \tilde{x} &= \arg\max_{z\in\mathcal{X}} \Big\{ \big(z P \mathrm{diag}(My_t) x^\top\big) \pi_{t-1}^z \Big\}, \end{aligned}$$

and the constant of proportionality for $\pi_t(x)$ is independent of $\{x_s\}_{s\le t}$.

Proof. Clearly, at time $t = 0$, before any observations have been made, we know $\chi_0^x = \{x\}$ and $\pi_0 = \mu_0$. At each t, we know from (5.3.1) that χ_t^x should be obtained by extending a path $\chi_{t-1}^{\tilde{x}}$, as the change in probability depends only on y, x_t, x_{t-1}, and all coefficients are positive. The result follows from our earlier calculations. □

Overall, this gives an easily-implementable (if tedious to compute by hand!) algorithm for decoding a signal observed in (possibly large amounts of) noise. The Viterbi algorithm is ubiquitous in applications.

Example 5.19. Using the same setup as the previous example, at time 1 we have

$$\pi_1^1 = \min\{0.63 \times 1, 0.36 \times 0\} = 0.63 \qquad \chi_1^1 = [1,1]$$
$$\pi_1^2 = \min\{0.06 \times 1, 0.12 \times 0\} = 0.06 \qquad \chi_1^2 = [1,2]$$

At time $t = 2$, we have

$$\pi_1^1 = \min\{0.07 \times 0.63, 0.04 \times 0.06\} = \min\{0.0441, 0.0024\} = 0.0441 \qquad \chi_1^1 = [1,1,1]$$
$$\pi_1^2 = \min\{0.24 \times 0.63, 0.48 \times 0.06\} = \min\{0.1512, 0.0288\} = 0.1512 \qquad \chi_1^2 = [1,1,2]$$

Repeating this, we get

Time	0	1	2	3	4
π^1	1	0.63	0.0441	0.00605	0.0043
π^2	0	0.06	0.1512	0.07258	0.0087
χ^1	1	[1,1]	[1,1,1]	[1,1,2,1]	[1,1,2,2,1]
χ^2	0	[1,2]	[1,1,2]	[1,1,2,2]	[1,1,2,2,2]

Observe that the most likely path for X up to time 4 does not end with the most likely state at time 4!

In its basic formulation, the Viterbi algorithm keeps $|\mathcal{X}|$ copies of the entire history of the path in memory. However, assuming our Markov chain is ergodic, we do not expect the value of x_t to have much impact on the most likely value of x_s for $s \ll t$. Therefore, we will often have that early sections of χ_t^x are the same for all values of x – these values will never change, and so can be sent to output (and hence do not need to be stored further).

The challenge in applications is that it requries a list of all codewords, the Markov transitions between them, and the emission matrix associated with the DMC, which limits the scale it can be applied to. Often these parameters need to be estimated, possibly with only limited observation of the underlying signal. This connects with the large area of stochastic filtering, which attempts to address these problems in wider contexts.

Appendix A

Some useful probability theory

We briefly recall and introduce basic notation from probability theory. We refer the reader to [3, 4] for an elementary introduction to probability theory and to [2, 5] for a more exhaustive treatment (or to the notes for B8.1).

A.1 Measure theory

A *measurable space* $(\mathcal{X}, \mathcal{A})$ consists of a set $\mathcal{X}$ and and a *σ-algebra* $\mathcal{A}$, that is a collection of subsets of $\mathcal{X}$ such that

(1) $\mathcal{X} \in \mathcal{A}$;

(2) $A \in \mathcal{A}$ implies $A^c \in \mathcal{A}$;

(3) if $A_n \in \mathcal{A}$ then $\cup_{n\in\mathbb{N}} A_n \in \mathcal{A}$.

Example A.1. Take $\mathcal{X} = \{a, b, c, d]\}$ and $\mathcal{A} = \{\emptyset, \{a, b\}, \{c, d\}, \{a, b, c, d\}\}$, then $(\mathcal{X}, \mathcal{A})$ is a discrete measurable space.

Take $\mathcal{X} = \mathbb{R}$ and $\mathcal{A}$ the smallest σ-algebra that contains all open sets ("Borel σ-algegra"), then $(\mathcal{X}, \mathcal{A})$ is one of the most often-used measurable spaces.

Given two measurable spaces $(\mathcal{X}_1, \mathcal{A}_1)$ and $\mathcal{X}_2, \mathcal{A}_2)$, we call a map $X : \mathcal{X}_1 \longrightarrow \mathcal{X}_2$ measurable with respect to $\mathcal{A}_1 \backslash \mathcal{A}_2$ if

$$X^{-1}(A) \in \mathcal{A}_1 \text{ for } \forall A \in \mathcal{A}_2.$$

It is a good exercise to show that the space of measurable maps (with respect to $\mathcal{A}_1 \backslash \mathcal{A}_2$) is closed under addition, scalar multiplication, $\liminf, \limsup$, etc.

A.2 Probability spaces

A *probability space* $(\Omega, \mathcal{F}, \mathbb{P})$ is a measurable space $(\Omega, \mathcal{F})$ together with a map $\mathbb{P} : \mathcal{F} \longrightarrow [0, 1]$ such that

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

(2) (σ-additivity) $\mathbb{P}(\cup_n A_n) = \sum \mathbb{P}(A_n)$ for disjoint $(A_n) \subset \mathcal{F}$ (i.e. $A_i \cap A_j = \emptyset$ for any $i \neq j$).

We refer to Ω as *sample space*, to elements of $\mathcal{F}$ as *events*, and to $\mathbb{P}$ as the *probability measure*. An $\mathcal{F}\backslash\mathcal{A}$-measurable map $X : \Omega \longrightarrow \mathcal{X}$ from Ω to another measurable space $\mathcal{X}$ with σ-algebra $\mathcal{A}$ is called a *random variable*, and $\mathcal{X}$ is called the *state space*.

Example A.2. A player flips a coin and wins one pound if it is a head, otherwise the player wins nothing. We can model this as follows: Let $\Omega = \{H, T\}, \mathcal{F} = \{\emptyset, \{H\}, \{T\}, \{H, T\}\}$, and

$$X(\omega) = \begin{cases} 1 & \text{if } \omega = H \\ 0 & \text{if } \omega = T \end{cases}.$$

Given any number $p \in [0, 1]$, we can define a probability measure by $\mathbb{P}(H) = p, \mathbb{P}(T) = 1 - p$[1] Notice that with different value $p \in [0, 1]$ we get different probability measure $\mathbb{P}$,

Example A.3. For an integer N, let $\Omega = \{H, T\}^N$ and $\mathcal{F}$ be the class of all subsets of Ω. Then $X_i(\omega) := \begin{cases} 1 & \text{if } \omega_i = H \\ 0 & \text{if } \omega_i = T \end{cases}$ is a random variable on $(\Omega, \mathcal{F})$ and so is

$$X_1 + \cdots + X_n$$

(the number of heads in n coin tosses).

We call two events $A, B \in \mathcal{F}$ *independent* events if $\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B)$. Otherwise, we call them dependent. Given two random variables on $(\Omega, \mathcal{F}, \mathbb{P})$ we say that X and Y are independent if $\{X \in A\}$[2] and $\{Y \in B\}$ are independent for all $A, B \in \mathcal{A}$. In the case of the discrete random variables, it is sufficient to require $\mathbb{P}(X = x, Y = y) = \mathbb{P}(X = x)\mathbb{P}(Y = y)$ for all $x \in X(\Omega), y \in Y(\Omega)$.

A.3 Discrete random variables

Throughout this course, we are mostly interested in random variables that take values in a countable set. More precisely, we call $X : \Omega \longrightarrow \mathbb{R}$ a discrete random variable, if the image $X(\Omega)$ is a countable subset of $\mathbb{R}$ and $X^{-1}(x) \in \mathcal{F}$ for all $x \in \mathbb{R}$. In this course, we often denote the image space of X with $\mathcal{X}$. Given a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ and a discrete random variable X, we call

$$p_X(x) := \mathbb{P}(X = x)$$

the probability mass function (pmf) of X (also distribution of X).

Example A.4. In Example A.2, X is a discrete random variable with the image space $X(\Omega) = \{0, 1\}$.

We can regard two discrete random variable X, Y with image spaces $\mathcal{X}, \mathcal{Y}$ as one discrete random varible (X, Y) with image space $\mathcal{X} \times \mathcal{Y}$. We call

$$p_{X,Y}(x, y) := \mathbb{P}(X = x, Y = y) = \mathbb{P}((X, Y) = (x, y))$$

[1] To be rigorous, we should write $\mathbb{P}(\{H\}) = p$, which is often simplified to the notation $\mathbb{P}(H) = p$.

[2] The rigorous expression for $\{X \in A\}$ is $\{\omega : X(\omega) \in A\}$.

the joint pmf of X and Y. Given a pmf on $\mathcal{X} \times \mathcal{Y}$ we call

$$p_X(x) := \sum_{y \in \mathcal{Y}} p_{X,Y}(x, y)$$

the marginal on $\mathcal{X}$, and the marginal on $\mathcal{Y}$ is defined similarly.

A.4 Expectation

Given a probaiblity space $(\Omega, \mathcal{F}, \mathbb{P})$ and a discrete random variable $X : \Omega \longrightarrow \mathcal{X} \subset \mathbb{R}$, we call

$$\mathbb{E}[X] := \sum_{x \in \mathcal{X}} x\mathbb{P}(X = x)$$

the expectation of X whenever this sum converges absolutely. If X and Y are discrete random variable defined on $(\Omega, \mathcal{F}, \mathbb{P})$, so are (X, Y) and any measurable function of (X, Y).

We call $\mathrm{Var}[X] := \mathbb{E}[(X - E[X])^2]$ the variance of X (if this expectation exists) and

$$\mathrm{Cov}[X, Y] := \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$$

the covariance of X and Y.

It is well-known that X and Y are independent (denoted as $X \perp Y$) iff

$$\mathbb{E}[f(X)g(Y)] = \mathbb{E}[f(X)]\mathbb{E}[g(Y)]$$

for all functions f, g for which the two expectations on the right hand side exists.

A.5 Conditional probabilities and conditional expectations

Given a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ and $A \in \mathcal{F}$ with $\mathbb{P}(A) > 0$, we define the conditional probability $\mathbb{P}(\cdot|A) : \mathcal{F} \longrightarrow [0, 1]$ as

$$\mathbb{P}(B|A) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(A)}.$$

Note that $(\Omega, \mathcal{F}, \mathbb{Q}_A)$ is a probability space with $\mathbb{Q}_A(\cdot) := \mathbb{P}(\cdot|A)$. Given two discrete random variables X and Y, we call

$$p_{Y|X}(y|x) := p_{Y|X=x}(y) := \mathbb{P}(Y = y|X = x) = \begin{cases} \frac{p_{X,Y}(x,y)}{p_X(x)} & \text{if } p_X(x) > 0 \\ 0 & \text{otherwise} \end{cases}$$

the conditional pmf of Y given X.

For $A \in \mathcal{F}$ and a discrete random variable X, define the conditional expectation of X given A as

$$\mathbb{E}[X|A] := \sum_{x \in \mathcal{X}} xP(X = x|A).$$

We often apply this with $A = \{Y = y\}$ where Y is another discrete random variable, i.e. $\mathbb{E}[X|A] = \mathbb{E}[X|Y = y]$.

Bibliography

[1] Thomas M. Cover and Joy A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[2] Richard M Dudley. *Real analysis and probability*, volume 74. Cambridge University Press, 2002.

[3] Geoffrey Grimmett and David Stirzaker. *Probability and random processes*. Oxford university press, 2001.

[4] Geoffrey Grimmett and Dominic Welsh. *Probability: an introduction*. Oxford University Press, 2014.

[5] Paul Malliavin. *Integration and probability*, volume 157. Springer Science & Business Media, 1995.