Information Theory

University of Amsterdam, Master of Logic, Fall 2016

Yfke Dulek and Christian Schaffner

The source is available on github and we are grateful for any feedback including typos and suggestions for improvements! Feel free to send us pull requests.

Parts of the text is based on, or taken verbatim, from 'The Mathematical Theory of Information, and Applications' (version 2.0) by Ronald Cramer and Serge Fehr [CF]. The lay-out is based on the Legrand Orange Book (version 2.1.1), licenced under the Creative Commons License BY-NC-SA 3.0.

Contents

Chapter 1: Probability and Entropy

1.1 Preliminaries: Probability Theory

For this course, we will only be concerned with discrete probabilities. This section formalizes some notions you should already be familiar with: probability spaces, events and probability distributions.

Definition 1.1.1 — Probability space. A (discrete) probability space $(\Omega, \mathcal{F}, P)$ consists of a discrete, non-empty *sample space* Ω, an *event space* $\mathcal{F} \subseteq \mathcal{P}(\Omega)$ and a *probability measure* P which is a function $P : \Omega \to \mathbb{R}_{\geq 0}$ that satisfies

$$\sum_{\omega \in \Omega} P(\omega) = 1.$$

The event space $\mathcal{F}$ is required to be non-empty and closed under intersection, union and complements. For convenience, we will most often assume that $\mathcal{F}$ equals the powerset $\mathcal{P}(\Omega)$ of Ω, i.e. it contains all possible subsets of events, and therefore fulfils the required properties.

Definition 1.1.2 — Event. An event $\mathcal{A}$ is an element of the event space $\mathcal{F} \subseteq \mathcal{P}(\Omega)$, i.e. a subset $\mathcal{A}$ of the sample space Ω. Its probability is defined as

$$P[\mathcal{A}] := \sum_{\omega \in \mathcal{A}} P(\omega),$$

where by convention $P[\emptyset] = 0$.

As a notational convention, we write $P[\mathcal{A}, \mathcal{B}]$ for $P[\mathcal{A} \cap \mathcal{B}]$, and $P[\overline{\mathcal{A}}]$ for $P[\Omega \setminus \mathcal{A}]$.

Exercise 1.1.1 Prove the following identities (for arbitrary events $\mathcal{A}, \mathcal{B} \subseteq \Omega$):

$$P[\overline{\mathcal{A}}] = 1 - P[\mathcal{A}] \tag{1.1}$$

$$P[\mathcal{A} \cup \mathcal{B}] = P[\mathcal{A}] + P[\mathcal{B}] - P[\mathcal{A}, \mathcal{B}] \tag{1.2}$$

$$P[\mathcal{A}] = P[\mathcal{A}, \mathcal{B}] + P[\mathcal{A}, \overline{\mathcal{B}}]. \tag{1.3}$$

It is often useful to consider the probability of an event *given* that some other event happened:

Definition 1.1.3 — Conditional probability. For events $\mathcal{A}$ and $\mathcal{B}$ with $P[\mathcal{A}] > 0$, the conditional probability of $\mathcal{B}$ given $\mathcal{A}$ is defined as

$$P[\mathcal{B}|\mathcal{A}] := \frac{P[\mathcal{A},\mathcal{B}]}{P[\mathcal{A}]}.$$

Example 1.1.1 — Fair die. We throw a six-sided fair die once, and consider the number that comes up. The sample space for this experiment is $\Omega = \{1,2,3,4,5,6\}$, with event space $\mathcal{F} = \mathcal{P}(\Omega)$ and probability measure $P[i] = \frac{1}{|\Omega|} = \frac{1}{6}$ for all $i \in \Omega$ (this is a **uniform** probability measure). Consider the events $\mathcal{A} = \{2,4,6\}$ and $\mathcal{B} = \{3,6\}$. Using the formulas in Definitions 1.1.2 and 1.1.3, we can compute the following probabilities:

$$\begin{aligned}
P[\mathcal{A}] &= \frac{1}{2} && \text{(the outcome is even)}\\
P[\mathcal{B}] &= \frac{1}{3} && \text{(the outcome is a multiple of 3)}\\
P[\mathcal{A},\mathcal{B}] &= P[\{6\}] = \frac{1}{6} && \text{(the roll is even \textit{and} a multiple of 3)}\\
P[\mathcal{A}|\mathcal{B}] &= \frac{1/6}{1/3} = \frac{1}{2} && \text{(the roll is even, \textit{given} that it is a multiple of 3)}\\
P[\mathcal{B}|\mathcal{A}] &= \frac{1/6}{1/2} = \frac{1}{3} && \text{(the roll is a multiple of 3, \textit{given} that it is even)}
\end{aligned}$$

This example shows that in general, $P[\mathcal{A}|\mathcal{B}]$ is *not equal* to $P[\mathcal{B}|\mathcal{A}]$.

Definition 1.1.4 — Discrete Random Variable (RV). Let $(\Omega,\mathcal{F},P)$ be a discrete probability space. A random variable X is a function $X : \Omega \to \mathcal{X}$ where $\mathcal{X}$ is a set, and we may assume it to be discrete.

A *real* random variable is one whose image is contained in $\mathbb{R}$. A (The *image* and the *range* of a random variable X are given by the image and the range of X in the function-theoretic sense.) The image of a *binary* random variable is a set $\{x_0,x_1\}$ with only two elements.

Definition 1.1.5 — Probability distribution. Let X be a random variable. The probability distribution of X is the function $P_X : \mathcal{X} \to [0,1]$ defined as

$$P_X(x) := P[X = x],$$

where $X = x$ denotes the event $\{\omega \in \Omega \,|\, X(\omega) = x\}$.

Alternatively, one can write $P_X(x) = P(X^{-1}(x))$ to express that the probability of x is precisely the P-measure of the pre-image of x under the random variable X.

Exercise 1.1.2 Verify that $(\mathcal{X},\mathcal{P}(\mathcal{X}),P_X)$ is itself a probability space. ■

We say that P_X is a **uniform** distribution if the associated probability measure is uniform, i.e. $P_X(x) = \frac{1}{|\mathcal{X}|}$. The **support** of a random variable or a probability distribution is defined as $\mathrm{supp}(P_X) := \{x \in \mathcal{X} \mid P_X(x) > 0\}$, the points of the range which have strictly positive probability. We often slightly abuse notation and write $\mathrm{supp}(X)$ instead.

When given two or more random variables defined on the same probability space, we can consider the probability that each of the variables take on a certain value:

Definition 1.1.6 — Joint probability distribution. Let X and Y be two random variables defined on the same probability space, with respective ranges $\mathcal{X}$ and $\mathcal{Y}$. The pair XY is a random variable with probability distribution $P_{XY} : \mathcal{X} \times \mathcal{Y} \to [0,1]$ given by

$$P_{XY}(x,y) := P[X = x, Y = y].$$

This definition naturally extends to three and more random variables. Unless otherwise stated, a collection of random variables is assumed to be defined on the same (implicit) probability space, so that their joint distribution is always well-defined.

If $P_{XY} = P_X \cdot P_Y$, in the sense that $P_{XY}(x,y) = P_X(x)P_Y(y)$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, then the random variables X and Y are said to be **independent**. If a set of variables $X_1, \ldots, X_n$ are all mutually independent and all have the same distribution (i.e. $P_{X_i} = P_{X_j}$ for all i, j), then they are **independent and identically distributed**, or **i.i.d.**

From a joint distribution, we can always find out the "original" (or **marginal**) distribution of one of the random variables (for example, X) by **marginalizing** out the variable that we want to discard (for example, Y):

$$P_X(x) = \sum_{y \in \mathcal{Y}} P_{XY}(x,y). \tag{1.4}$$

This marginalization process also works with more than two random variables.

Like events, probability distributions can also be conditioned on probabilistic events:

Definition 1.1.7 — Conditional probability distribution. If $\mathcal{A}$ is an event with $P[\mathcal{A}] > 0$, then the conditional probability distribution of X given $\mathcal{A}$ is given by

$$P_{X|\mathcal{A}}(x) = \frac{P[X = x, \mathcal{A}]}{P[\mathcal{A}]}.$$

If Y is another random variable and $P_Y(y) > 0$, then we write

$$P_{X|Y}(x|y) := P_{X|Y=y}(x) = \frac{P_{XY}(x,y)}{P_Y(y)}$$

for the conditional distribution of X, given $Y = y$.

Note that again, both $(\mathcal{X}, P_{X|\mathcal{A}})$ and $(\mathcal{X}, P_{X|Y=y})$ themselves form probability spaces. Note also that if X and Y are independent, then

$$P_{X|Y}(x|y) = \frac{P_{XY}(x,y)}{P_Y(y)} = \frac{P_X(x) \cdot P_Y(y)}{P_Y(y)} = P_X(x), \tag{1.5}$$

which aligns well with our intuition of independent variables: the distribution of X remains unchanged when Y is fixed to a specific value.

Example 1.1.2 — Fair die (continued). Consider again the throw of a six-sided fair die as in Example 1.1.1. Let the random variable X describe the number of integer divisors for the outcome, that is

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

X is a real random variable, with range $\mathcal{X} = \{1,2,3\}$. The associated probability distribution is

$$P_X(1) = P[\{1\}] = \frac{1}{6}, \qquad P_X(2) = P[\{2,3,5\}] = \frac{1}{2}, \qquad P_X(3) = P[\{4,6\}] = \frac{1}{3}.$$

If we now condition on the event $\mathcal{A} = \{2,4,6\}$ (the outcome being even), we get that

$$P_{X|\mathcal{A}}(1) = 0, \qquad P_{X|\mathcal{A}}(2) = \frac{1}{3}, \qquad P_{X|\mathcal{A}}(3) = \frac{2}{3}.$$

If X is a random variable and $f : \mathcal{X} \to \mathcal{Y}$ is a surjective function, then $f(X)$ is a random variable, defined by composing the map f with the map X. Its image is $\mathcal{Y}$. Clearly,

$$P_{f(X)}(y) = \sum_{x \in \mathcal{X} : f(x) = y} P_X(x). \tag{1.6}$$

For example, $1/P_X(X)$ denotes the real random variable obtained from another random variable X by composing with the map $1/P_X$ that assigns $1/P_X(x) \in \mathbb{R}$ to $x \in \mathcal{X}$.

Definition 1.1.8 — Expectation. The expectation of a *real* random variable X is defined as

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

Note that if X is not real, then we can still consider the expectation of some function $f : \mathcal{X} \to \mathbb{R}$, where

$$\mathbb{E}[f(X)] = \sum_{x \in \mathcal{X}} P_X(x) \cdot f(X). \tag{1.7}$$

Definition 1.1.9 — Variance. The variance of a *real* random variable X is defined as

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

The variation is a measure for the deviation of the mean. Hoeffding's inequality (here stated for binary random variables) states that for a list of i.i.d. random variables, the average of the random variables is close to the expectation, except with very small probability. We state it here without proof.

Theorem 1.1.1 — Hoeffding's inequality. Let $X_1, \ldots, X_n$ be independent and identically distributed binary random variables with $P_{X_i}(0) = 1 - \mu$ and $P_{X_i}(1) = \mu$, and thus $\mathbb{E}[X_i] = \mu$. Then, for any $\delta > 0$

$$P\left[\sum_i X_i > (\mu + \delta) \cdot n\right] \leq \exp(-2\delta^2 n).$$

1.2 Some Important Distributions

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

has the **binomial(n, p)** distribution:

$$P_S(k) = \binom{n}{k} p^k (1-p)^{n-k} \qquad \text{where } k = 0, 1, 2, \ldots, n\,. \tag{1.8}$$

From simple properties of the expected value and variance, one can show that $\mathbb{E}[S] = np$ and $\text{Var}[S] = np(1-p)$.

- The **geometric(p)** distribution of a random variable Y is defined as the number of times one has to flip a Bernoulli(p) coin before it lands heads:

$$P_Y(k) = (1-p)^{k-1} p \qquad \text{where } k = 1, 2, 3, \ldots\,. \tag{1.9}$$

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

$$P_Z(k) = (1-p)^k p \qquad \text{where } k = 0, 1, 2, 3, \ldots\,. \tag{1.10}$$

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

1.3 Jensen's Inequality

In the following, let $\mathcal{D}$ be an interval in $\mathbb{R}$.

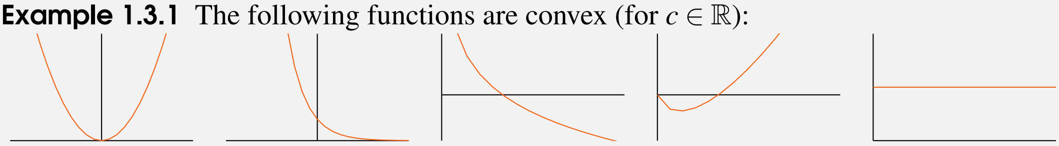
Definition 1.3.1 — Convex and concave functions. The function $f : \mathcal{D} \to \mathbb{R}$ is convex if for all $x_1, x_2 \in \mathcal{D}$ and all $\lambda \in [0,1] \subset \mathbb{R}$:

$$\lambda f(x_1) + (1-\lambda) f(x_2) \geq f(\lambda x_1 + (1-\lambda) x_2).$$

The function f is *strictly* convex if equality only holds when $\lambda \in \{0,1\}$ or when $x_1 = x_2$. The function f is (strictly) concave if the function $-f$ is (strictly) convex.

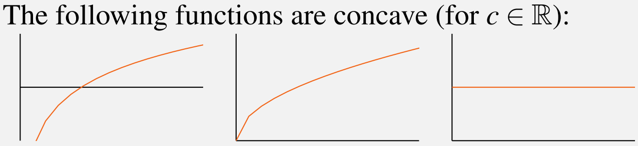
Intuitively, a function is convex if any straight line drawn between two points $f(x_1)$ and $f(x_2)$ lies above the graph of f entirely. For a concave function, such a line must lie entirely beneath the graph.

Example 1.3.1 The following functions are convex (for $c \in \mathbb{R}$):



x^2 $\quad e^{-x}$ $\quad \log(1/x)$ $\quad x \cdot \log x$ $\quad c$

The following functions are concave (for $c \in \mathbb{R}$):



$\log x$ $\quad \sqrt{x}$ $\quad c$

The following establishes a more formal method of proving the convexity of a function.

Proposition 1.3.1 Let $f : \mathcal{D} \to \mathbb{R}$. If $\mathcal{D}$ is open, and for all $x \in \mathcal{D}$, the second order derivative $f''(x)$ exists and is non-negative (positive), then f is convex (strictly convex).

We omit the proof, which can be found in, for example, [CF] (Lemma 1).

Theorem 1.3.2 — Jensen's inequality. Let $f : \mathcal{D} \to \mathbb{R}$ be a convex function, and let $n \in \mathbb{N}$. Then for any $p_1, \ldots, p_n \in \mathbb{R}_{\geq 0}$ such that $\sum_{i=1}^n p_i = 1$ and for any $x_1, \ldots, x_n \in \mathcal{D}$ it holds that

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

If f is strictly convex and $p_1, \ldots, p_n > 0$, then equality holds iff $x_1 = \cdots = x_n$.

In particular, if X is a real random variable whose image $\mathcal{X}$ is contained in $\mathcal{D}$, then

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

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

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

$$\begin{aligned}
\sum_{i=1}^{n} p_i f(x_i) &= p_n f(x_n) + \sum_{i=1}^{n-1} p_i f(x_i) \\
&= p_n f(x_n) + (1 - p_n) \sum_{i=1}^{n-1} \frac{p_i}{1 - p_n} f(x_i) \\
&\geq p_n f(x_n) + (1 - p_n) f\left(\sum_{i=1}^{n-1} \frac{p_i}{1 - p_n} x_i\right) && \text{(induction hypothesis)} \\
&\geq f\left(p_n x_n + (1 - p_n) \sum_{i=1}^{n-1} \frac{p_i}{1 - p_n} x_i\right) && \text{(definition of convexity)} \\
&= f\left(p_n x_n + \sum_{i=1}^{n-1} p_i x_i\right) \\
&= f\left(\sum_{i=1}^{n} p_i x_i\right). && (1.11)
\end{aligned}$$

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

1.4 Shannon Entropy

In this section, we explore a measure for the amount of uncertainty of random variables. Consider some probabilistic event $\mathcal{A}$ that occurs with probability $P[\mathcal{A}]$ for some probability measure P. The **surprisal value** $\log \frac{1}{P[\mathcal{A}]}$ indicates how surprised we should be when the event $\mathcal{A}$ occurs: events with small probabilities yield high surprisal values, and vice versa. An event that occurs with certainty ($P_X(\mathcal{A}) = 1$) yields a surprisal value of 0. For a random variable X, we consider the *expected* surprisal value to be an indicator of how much uncertainty is contained in the variable, or how much information is gained by revealing the outcome. This expected surprisal value is more commonly known as the (Shannon) entropy[1] of a random variable:

Definition 1.4.1 — Entropy. Let X be a random variable with image $\mathcal{X}$. The (Shannon) entropy $H(X)$ of X is defined as

$$H(X) := \mathbb{E}\left[\log \frac{1}{P_X(X)}\right] = \sum_{x\in\mathcal{X}} P_X(x) \cdot \log \frac{1}{P_X(x)} = -\sum_{x\in\mathcal{X}} P_X(x) \cdot \log P_X(x),$$

with the convention that the log function represents the *binary* logarithm $\log_2$. As another convention, for $x \in \mathcal{X}$ with $P_X(x) = 0$, the corresponding argument in the summation is declared 0 (which is justified by taking a limit).

It is important to realize that the entropy of X is a function (solely) of the *distribution* P_X of X. However, it is customary to write $H(X)$ instead of the formally correct $H(P_X)$.

Exercise 1.4.1 Prove that $\lim_{p\to 0} p\log(p) = 0$. ■

Proposition 1.4.1 — Positivity. Let X be a random variable with image $\mathcal{X}$. Then

$$0 \leq H(X) \leq \log(|\mathcal{X}|).$$

Equality on the left-hand side holds iff there exists $x \in \mathcal{X}$ with $P_X(x) = 1$ (and thus $P_X(x') = 0$ for all $x' \neq x$). Equality on the right-hand side holds iff $P_X(x) = 1/|\mathcal{X}|$ for all $x \in \mathcal{X}$.

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

$$H(X) = \sum_{x\in\mathcal{X}} P_X(x) \cdot \log \frac{1}{P_X(x)} \leq \log\left(\sum_{x\in\mathcal{X}} 1\right) = \log(|\mathcal{X}|). \tag{1.12}$$

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

Finally, for the characterization of the lower bound, it is obvious that $H(X) = 0$ if $P_X(x) = 1$ for some x, and, on the other hand, if $H(X) = 0$ then for any x with $P_X(x) > 0$ it must be that $\log(1/P_X(x)) = 0$ and hence $P_X(x) = 1$. ■

For a binary random variable X with image $\mathcal{X} = \{x_0, x_1\}$ and probabilities $P_X(x_0) = p$ and $P_X(x_1) = 1 - p$, we can write $H(X) = h(p)$, where h denotes the binary entropy function:

Definition 1.4.2 — Binary entropy function h. The binary entropy function is defined for $0 < q < 1$ as

$$h(q) := q\log\frac{1}{q} + (1-q)\log\frac{1}{1-q},$$

and is defined as $h(q) = 0$ for $q = 0$ or $q = 1$. The graph of h on the interval $[0, 1]$ is shown in Figure 1.1.

This binary entropy function is used, for example, to measure the entropy of a biased coin flip.

[1] Shannon once said: *My greatest concern was what to call it. I thought of calling it* information, *but the word was overly used, so I decided to call it* uncertainty. *When I discussed it with John von Neumann, he had a better idea. Von Neumann told me: "You should call it* entropy, *for two reasons. In the first place, your uncertainty function has been*

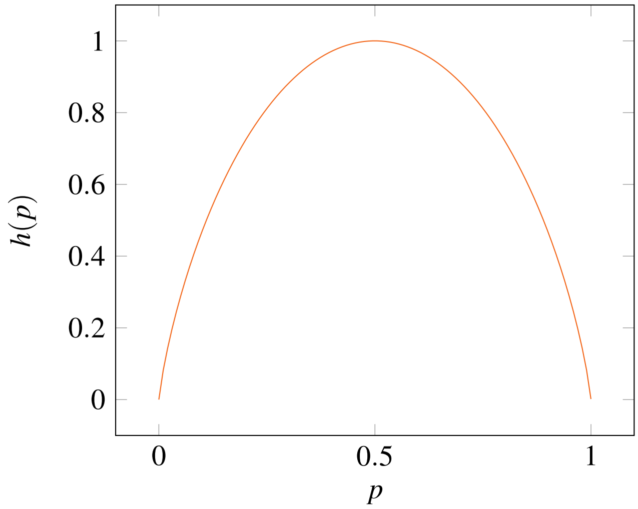


Figure 1.1: The binary entropy function h as a function of the probability p.

Example 1.4.1 Consider a random variable X with $\mathcal{X} = \{a,b,c\}$ and $P_X(a) = \frac{1}{2}$, $P_X(b) = P_X(c) = \frac{1}{4}$. The entropy of X is

$$H(X) = \frac{1}{2}\log 2 + \frac{1}{4}\log 4 + \frac{1}{4}\log 4 = \frac{1}{2} + \frac{1}{2} + \frac{1}{2} = \frac{3}{2}. \tag{1.13}$$

Another approach to computing the entropy of X by coming up with an appropriate underlying probability space (Ω, P): we toss a fair coin twice, giving $\Omega = \{\mathtt{hh}, \mathtt{ht}, \mathtt{th}, \mathtt{tt}\}$ and $P(\omega) = \frac{1}{4}$ for all $\omega \in \Omega$. The we define the function $X : \Omega \to \mathcal{X}$ as

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

This yields the correct distribution P_X. The following computation now leads to the entropy of X:

$$H(X) = h\left(\frac{1}{2}\right) + \frac{1}{2}h(0) + \frac{1}{2}h\left(\frac{1}{2}\right) = \frac{3}{2}. \tag{1.14}$$

The first coin toss determines whether the outcome is a (on heads h) or something else (on tails t). On heads, the second coin toss does not give any more information, whereas on tails, the second coin toss still decides between outcome b and outcome c. In general, the entropy of a random variable with probabilities $p_1, \ldots, p_n$ can be expressed as

$$\begin{aligned} H(p_1, \ldots, p_k, p_{k+1}, \ldots, p_n) = h(p_1 + \cdots + p_k) + &\\ &(p_1 + \cdots + p_k) H\left(\frac{p_1}{p_1 + \cdots + p_k} + \cdots + \frac{p_k}{p_1 + \cdots + p_k}\right) + \\ &(p_{k+1} + \cdots + p_n) H\left(\frac{p_{k+1}}{p_{k+1} + \cdots + p_n} + \cdots + \frac{p_n}{p_{k+1} + \cdots + p_n}\right). \end{aligned} \tag{1.15}$$

used in statistical mechanics under that name, so it already has a name. In the second place, and more important, nobody knows what entropy really is, so in a debate you will always have the advantage."

1.5 Conditional Entropy

Let X be a random variable and $\mathcal{A}$ an event. Applying Definition 1.4.1 to the conditional probability distribution $P_{X|\mathcal{A}}$ allows us to naturally define the entropy of X conditioned on the event $\mathcal{A}$, which leads to the following notion:

Definition 1.5.1 — Conditional entropy. Let X and Y be random variables, with respective images $\mathcal{X}$ and $\mathcal{Y}$. The conditional entropy $H(X|Y)$ of X given Y is defined as

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

with the convention that the corresponding argument in the summation is 0 for $y\in\mathcal{Y}$ with $P_Y(y)=0$, and where

$$H(X|\mathcal{A}) := \sum_{x\in\mathcal{X}} P_{X|\mathcal{A}}(x)\cdot\log\frac{1}{P_{X|\mathcal{A}}(x)}.$$

Note that conditional entropy $H(X|Y)$ is not the entropy of a probability distribution but an expectation: the average uncertainty about X when given Y. The following bound expresses that (on average!) additional information, i.e. knowing Y, can only *decrease* the uncertainty.

Proposition 1.5.1 Let X and Y be random variables with respective images $\mathcal{X}$ and $\mathcal{Y}$. Then

$$0\le H(X|Y)\le H(X)$$

Equality on the left-hand side holds iff X is determined by Y, i.e., for all $y\in\mathcal{Y}$, there is an $x\in\mathcal{X}$ such that $P_{X|Y}(x|y)=1$. Equality on the right-hand side holds iff X and Y are independent.

Proof. The lower bound follows trivially from the definition and from Proposition 1.4.1, and so does the characterization of when $H(X|Y)=0$. For the upper bound, note that

$$H(X|Y) = \sum_y P_Y(y)\sum_x P_{X|Y}(x|y)\log\frac{1}{P_{X|Y}(x|y)} = \sum_{x,y} P_{XY}(x,y)\log\frac{P_Y(y)}{P_{XY}(x,y)} \tag{1.16}$$

and

$$H(X) = \sum_x P_X(x)\log\frac{1}{P_X(x)} = \sum_{x,y} P_{XY}(x,y)\log\frac{1}{P_X(x)} \tag{1.17}$$

where the last equality is derived by marginalization. Note that in both expressions, we may restrict the sum to those pairs (x,y) with $P_{XY}(x,y)>0$. Using Jensen's inequality, it follows that

$$\begin{aligned} H(X|Y)-H(X) &= \sum_{x,y} P_{XY}(x,y)\log\frac{P_X(x)P_Y(y)}{P_{XY}(x,y)} \\ &\le \log\Big(\sum_{x,y} P_X(x)P_Y(y)\Big) \le \log\Big(\big(\sum_x P_X(x)\big)\big(\sum_y P_Y(y)\big)\Big) = \log 1 = 0\,. \end{aligned} \tag{1.18}$$

Note that in the second inequality, we replaced the summation over all (x,y) with $P_{XY}(x,y)>0$ by the summation over all $(x,y)\in\mathcal{X}\times\mathcal{Y}$. Inequality then follows by the monotonicity of the logarithm function.

For the first inequality, equality holds if and only if $P_{XY}(x,y)=P_X(x)P_Y(y)$ for all (x,y) with $P_{XY}(x,y)>0$, and for the second inequality, equality holds if and only if $P_{XY}(x,y)=0$ implies $P_X(x)P_Y(y)=0$ for any $x\in\mathcal{X}$ and $y\in\mathcal{Y}$. It follows that $H(X|Y)=H(X)$ if and only if $P_{XY}(x,y)=P_X(x)P_Y(y)$ for all $(x,y)\in\mathcal{X}\times\mathcal{Y}$. ■

Proposition 1.5.2 — Chain Rule. Let X and Y be random variables. Then

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

Proof. The chain rule is a simple matter of rewriting:

$$\begin{aligned}
H(XY) &= -\sum_{x,y} P_{XY}(x,y) \log P_{XY}(x,y) \\
&= -\sum_{x,y} P_{XY}(x,y) \log\big(P_X(x) P_{Y|X}(y|x)\big) \\
&= -\sum_{x,y} P_{XY}(x,y) \log P_X(x) - \sum_{x,y} P_{XY}(x,y) \log P_{Y|X}(y|x) \\
&= -\sum_{x} P_X(x) \log P_X(x) - \sum_{x} P_X(x) \sum_{y} P_{Y|X}(y|x) \log P_{Y|X}(y|x) \\
&= H(X) + H(Y|X). \qquad (1.19)
\end{aligned}$$

This was to be shown. ■

The following inequality, also known as the 'independence bound', follows from the fact that $H(Y|X) \leq H(Y)$:

Corollary 1.5.3 — Subadditivity.

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

Equality holds iff X and Y are independent.

Note that applying Definition 1.5.1 to the conditional distribution $P_{XY|\mathcal{A}}$ naturally defines $H(X|Y,\mathcal{A})$, the entropy of X given Y and conditioned on the event $\mathcal{A}$. Since the entropy is a function of the distribution of a random variable, the chain rule also holds when conditioning on an event $\mathcal{A}$. Furthermore, it holds that

$$H(X|YZ) = \sum_{z} P_Z(z) H(X|Y, Z{=}z), \qquad (1.20)$$

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

Corollary 1.5.4 Let X, Y and Z be random variables. Then

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

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

$$H(X_1 \cdots X_n) = H(X_1) + H(X_2|X_1) + \cdots + H(X_n|X_{n-1} \cdots X_1). \qquad (1.21)$$

Example 1.5.1 Consider the binary random variables X and Y, with joint distribution

$$P_{XY}(00) = \frac{1}{2}, \quad P_{XY}(01) = \frac{1}{4}, \quad P_{XY}(10) = 0, \quad P_{XY}(11) = \frac{1}{4}.$$

By marginalization, we find that $P_X(0) = \frac{3}{4}$ and $P_X(1) = \frac{1}{4}$, while $P_Y(0) = P_Y(1) = \frac{1}{2}$. This allows us to make the following computations:

$$H(XY) = \frac{1}{2}\log 2 + \frac{1}{4}\log 4 + \frac{1}{4}\log 4 = \frac{3}{2} \tag{1.22}$$

$$H(X) = h\left(\frac{1}{4}\right) = h\left(\frac{3}{4}\right) \approx 0.81 \tag{1.23}$$

$$H(Y) = h\left(\frac{1}{2}\right) = 1 \tag{1.24}$$

$$H(X|Y) = H(XY) - H(Y) = \frac{1}{2} \tag{1.25}$$

$$H(Y|X) = H(XY) - H(X) \approx 0.69 \tag{1.26}$$

We also could have computed $H(X|Y)$ and $H(Y|X)$ directly through the definition of conditional entropy.

Note that for this specific distribution, $H(X|Y=1) > H(X)$. It is important to remember that Proposition 1.5.1 only holds on average, not for specific values of Y. Note also that in this example, $H(X|Y) \neq H(Y|X)$.

1.6 Mutual Information

Definition 1.6.1 — Mutual information. Let X and Y be random variables. The mutual information $I(X;Y)$ of X and Y is defined as

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

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

$$I(X;Y) = H(X) + H(Y) - H(XY) \qquad \text{(by chain rule)} \tag{1.27}$$

$$I(X;Y) = I(Y;X) \qquad \text{(“symmetry")} \tag{1.28}$$

$$I(X;Y) \geq 0 \qquad \text{(by subadditivity)} \tag{1.29}$$

$$I(X;Y) = 0 \text{ iff } X \text{ and } Y \text{ are independent} \tag{1.30}$$

$$I(X;X) = H(X) \qquad \text{(“self-information”)} \tag{1.31}$$

Applying Definition 1.6.1 to the conditional distribution $P_{XY|\mathcal{A}}$ naturally defines $I(X;Y|\mathcal{A})$, the mutual information of X and Y conditioned on the event $\mathcal{A}$.

Definition 1.6.2 — Conditional mutual information. Let X,Y,Z be random variables. Then the conditional mutual information of X and Y given Z is defined as

$$I(X;Y|Z) = \sum_z P_Z(z) I(X;Y|Z=z),$$

with the convention that the corresponding argument in the summation is 0 for z with $P_Z(z) = 0$.

Conditional mutual information has properties similar to the ones we saw above:

$$I(X;Y|Z) = I(Y;X|Z) \tag{1.32}$$

$$I(X;Y|Z) \geq 0 \tag{1.33}$$

$$I(X;Y|Z) = 0 \text{ iff } X \text{ and } Y \text{ are independent given } Z \tag{1.34}$$

Furthermore, the previous bounds $H(X) \geq 0$, $H(X|Y) \geq 0$, and $I(X;Y) \geq 0$, can all be seen as special cases of $I(X;Y|Z) \geq 0$. These bounds, and any bound they imply, are called **Shannon inequalities**.

It is important to realize that $I(X;Y|Z)$ may be larger or smaller than (or equal to) $I(X;Y)$. The following is easy to verify (and is sometimes used as definition of $I(X;Y|Z)$).

Proposition 1.6.1 Let X,Y,Z be random variables. Then

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

By this result, we obtain:

Corollary 1.6.2 — Chain rule for mutual information. Let W,X,Y and Z be random variables. Then

$$I(WX;Y|Z) = I(X;Y|Z) + I(W;Y|ZX).$$

Proof. The proof is a matter of writing out definitions and applying the generalized chain rule.

$$\begin{aligned} I(WX;Y|Z) &= H(WX|Z) - H(WX|YZ) \\ &= (H(X|Z) + H(W|XZ)) - (H(X|YZ) + H(W|XYZ)) \\ &= H(X|Z) - H(X|YZ) + H(W|XZ) - H(W|XYZ) \\ &= I(X;Y|Z) + I(W;Y|XZ). \end{aligned} \tag{1.35}$$

■

1.7 Relative entropy

A measure that is related to the mutual information is the relative entropy: it reflects how different two distributions are:

Definition 1.7.1 — Relative entropy. The relative entropy (or: **Kullback-Leibler divergence**) of two probability distributions P and Q over the same $\mathcal{X}$ is defined by

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

where by convention, $\log \frac{p}{0} = \infty$ for all p.

Note that if $Q(x) = 0$ for some x with $P(x) > 0$, then $D(P||Q) = \infty$.

Exercise 1.7.1 Show that $I(X;Y) = D(P_{XY}||P_X \cdot P_Y)$. ■

This exercise, combined with the equality condition in Theorem 1.7.1 below, shows that the mutual information is a measure of 'how independent' the variables X and Y are: if $P_{XY} = P_X \cdot P_Y$, the variables are independent and their mutual information is zero.

Theorem 1.7.1 — Information inequality. For any two probability distributions P and Q defined on the same $\mathcal{X}$,

$$D(P||Q) \geq 0.$$

Equality holds if and only if $P = Q$.

Proof. Left as an exercise. Hint: use Jensen's inequality. ■

Even though relative entropy is always nonnegative, it is not a proper distance measure, because it is not symmetric and does not satisfy the triangle inequality.

1.8 Entropy Diagrams

We finish this chapter by visually summing up the relations between entropy, joint entropy, conditional entropy, mutual information, and conditional mutual information. For two and three random variables, the relations between these different information-theoretic measures can be nicely represented by means of a Venn-diagram-like **entropy diagram**. The case of two random variables is illustrated in Figure 1.2 (left). From the diagram, one can for instance easily read off the relations $H(X|Y) \leq H(X)$, $I(X;Y) = H(X) + H(Y) - H(XY)$ etc. The case of three random variables is illustrated in Figure 1.2 (right). Also here, one can easily read off all the relations between the information-theoretic measures, like for instance $H(X|YZ) = H(X) - I(X;Z) - I(X;Y|Z)$, which is a relation that is otherwise not immediately obvious.

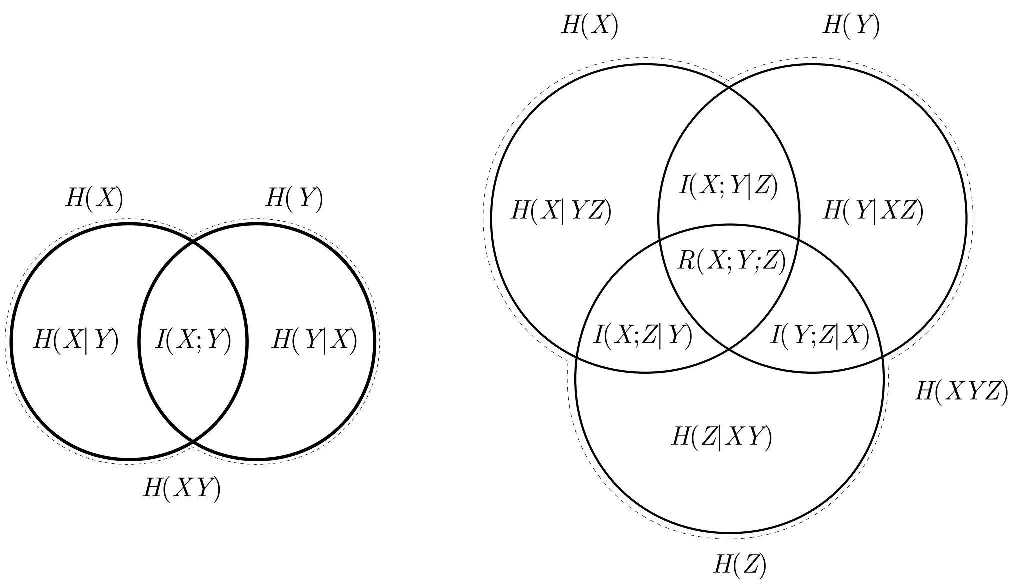


Figure 1.2: Entropy diagram for two (left) and three (right) random variables. The areas encompassed by the dotted lines represent $H(XY)$ and $H(XYZ)$, respectively.

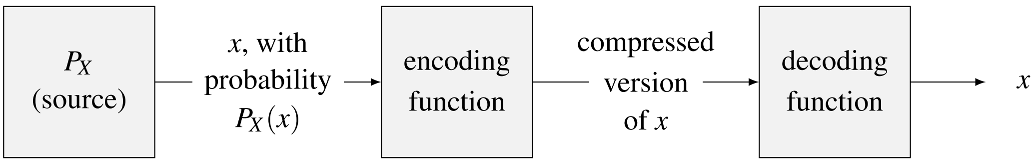
One subtlety with the entropy diagram for three random variables is that the "area in the middle", $R(X;Y;Z) = I(X;Y) - I(X;Y|Z)$, may be *negative*.

1.9 Further Reading

- Sections 2.1, 2.2, 3.1-3.3 of [CF]
- Sections 2.1, 2.2, 2.6 of [CT]
- For more background on probability theory, check for instance the lecture script of the Master of Logic course "Basic Probability:Theory" by Philip Schulz and Christian Schaffner.

Chapter 2: Source Coding

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



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

We explore both **lossless** codes (where we want to recover the original data with certainty) and **lossy** codes (where with small probability, the data is lost).

2.1 Symbol Codes

We start by investigating codes that encode a source one symbol at a time. Later on, we will also see codes that group the source symbols together into blocks.

Definition 2.1.1 — Binary symbol code. Let P_X be the distribution of a random variable X (with image $\mathcal{X}$). A binary symbol code for P_X is an injective function $C : \mathcal{X} \to \{0,1\}^*$.

The **extended** code $C^* : \mathcal{X}^* \to \{0,1\}^*$ is defined by concatenation:

$$C^*(x_1,...,x_n) := C(x_1)|\cdots|C(x_n).$$

Here, $\mathcal{X}^* = \bigcup_{n\in\mathbb{N}} \mathcal{X}^n \cup \{\perp\}$, and $\perp$ is the empty string.

We often refer to the set of codewords, $\mathcal{C} = \mathrm{im}(C)$, as code and leave the actual encoding function C implicit.

The injectivity of C ensures that we can always uniquely decode $C(x)$. However, if one transmits a sequence $x_1,\ldots,x_m \in \mathcal{X}$ (or stores them "sequentially") by sending the concatenation $C(x_1,...,x_n)$, ambiguities may arise, namely in cases where it is possible to parse this long string in two consistent but different ways. Indeed, injectivity of the encoding function per se does not rule out that there exists a positive integer m' and elements $x'_1,\ldots,x'_{m'} \in \mathcal{X}$ such that $C(x_1)|\cdots|C(x_m) = C(x'_1)|\cdots|C(x'_{m'})$. Of course, this problem can be circumvented by introducing a special separation symbol. However, such a symbol might not be available, and maybe even more importantly, even if an additional symbol *is* available, then one can often create a better code by using it as an ordinary code symbol (in addition to 0 and 1) rather than as a special separation symbol. This is why it is interesting to study the following class of symbol codes:

Definition 2.1.2 — Uniquely decodable code. A binary symbol code $C : \mathcal{X} \to \{0,1\}^*$ is uniquely decodable if C^* is injective as well.

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

Definition 2.1.3 — Prefix-free code. A binary symbol code $C : \mathcal{X} \to \{0,1\}^*$ is prefix-free (or: **instantaneous**) if for all $x, x' \in \mathcal{X}$ with $x \neq x'$, $C(x)$ is *not* a prefix of $C(x')$.

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

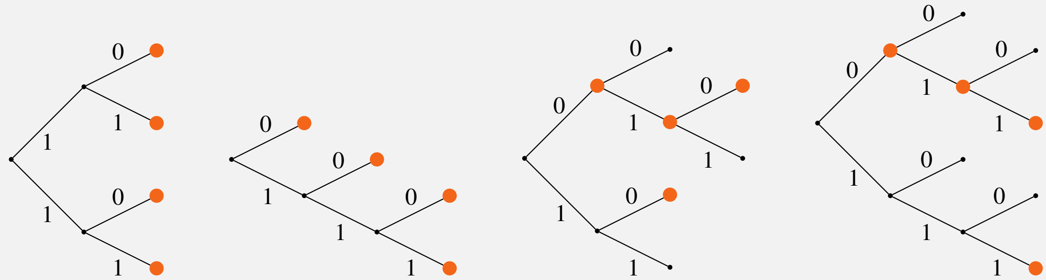
Proposition 2.1.1 If a code $\mathcal{C}$ is prefix-free and $\mathcal{C} \neq \{\perp\}$ then $\mathcal{C}$ is uniquely decodable.

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

Example 2.1.1 The following are three codes for the source P_X, with $\mathcal{X} = \{a,b,c,d\}$:

x	$P_X(x)$	$C_1(x)$	$C_2(x)$	$C_3(x)$	$C_4(x)$
a	1/2	00	0	0	0
b	1/4	01	10	010	01
c	1/8	10	110	01	011
d	1/8	11	111	10	111

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



C_1 C_2 C_3 C_4

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

For efficiency reasons, we are often interested in the average (expected) length of a code C:

Definition 2.1.4 — Average length. Let $\ell(s)$ denote the length of a string $s \in \{0,1\}^*$. The (average) length of a code C for a source P_X is defined as

$$\ell_C(P_X) := \mathbb{E}[\ell(C(X))] = \sum_{x\in\mathcal{X}} P_X(x)\ell(C(x)) .$$

Example 2.1.2 For the codes from Example 2.1.1, we obtain the following average codeword lengths: $\ell_{C_1}(P_X) = 2, \ell_{C_2}(P_X) = \ell_{C_4}(P_X) = \frac{1}{2}\cdot 1 + \frac{1}{4}\cdot 2 + \frac{1}{8}\cdot 3 + \frac{1}{8}\cdot 3 = \frac{7}{4} = 1.75$ and $\ell_{C_3}(P_X) = \frac{1}{2}\cdot 1 + \frac{1}{4}\cdot 3 + \frac{1}{8}\cdot 2 + \frac{1}{8}\cdot 2 = \frac{7}{4} = 1.75$. We see that the codes C_2, C_3, C_4 have a smaller average codeword length, but C_2 and C_4 are preferred over C_3 because their unique decodability.

Notice that the individual codeword lengths of codes C_2 and C_4 correspond exactly to the surprisal values of P_X in bits, e.g. $\ell(C_2(b)) = \ell(C_4(b)) = 2 = -\log P_X(b)$. Therefore, the computations of the entropy $H(X)$ and of the average code length $\ell_{C_2}(P_X)$ are exactly the same, and we have that $H(X) = \ell_{C_2}(P_X) = \ell_{C_4}(P_X)$. We will see in Section 2.3 below that this property characterizes optimal codes.

Definition 2.1.5 — Minimal code length. The minimal code length of a source P_X is defined as

$$\ell_{\min}(P_X) := \min_{C\in\mathfrak{C}} \ell_C(P_X)$$

where $\mathfrak{C}$ is some class of codes, for example the set of all prefix-free codes (resulting in $\ell_{\min}^{\text{p.f.}}$), or the set of all uniquely decodable codes (resulting in $\ell_{\min}^{\text{u.d.}}$).

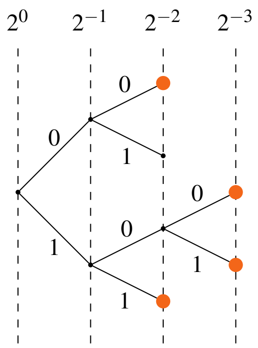
2.2 Kraft's Inequality

As argued, prefix-freeness is a nice feature, but it is also considerably more restrictive than mere unique decodability; thus, it is natural to ask: how much do we lose (in terms of the average codeword length) by requiring the encoding to be prefix-free rather than merely uniquely decodable? Surprisingly, the answer is: *nothing*. In this section, we will show that the length of an optimal prefix-free code and the length of an optimal uniquely decodable code coincide. In the next section, we will see that these lengths are essentially given by the Shannon entropy.

Theorem 2.2.1 — Kraft's inequality. There exists a prefix-free code with image $\mathcal{C} = \{c_1, \ldots, c_m\}$ and codeword lengths $\ell_i := \ell(c_i)$, if and only if

$$\sum_{i=1}^{m} 2^{-\ell_i} \leq 1.$$

Proof. For the forward direction, suppose we have a prefix-free code C with image $\mathcal{C} = \{c_1, \ldots, c_m\}$ and codeword lengths $\ell_i := \ell(c_i)$. View this code as a tree, with codewords only on the leaves (but not necessarily all the leaves), and assign a weight of 2^{-d} to every node in the tree at depth d (including the leaves):



Note that the weight of each node is exactly the sum of the weight of its direct children, and thereby that the weight of the root is exactly the weight of all of the leaves. Since every codeword c_i resides on a leaf of depth ℓ_i (but not all leaves are necessarily occupied), the weight of the root is *at least* the sum of all the codeword weights:

$$\sum_{i=1}^{m} 2^{-\ell_i} \leq 2^0 = 1. \tag{2.1}$$

For the backward direction, we build a code $\mathcal{C} = \{c_1, \ldots, c_m\}$ with $\ell(c_i) = \ell_i$ by selecting the appropriate leaves of a binary tree as codewords, assigning the most 'expensive' (i.e. those with small depth) first. We proceed by induction on the number of codewords, m:

For $m = 1$, the construction is clear: we can assign any string of length ℓ_1 to represent the single codeword.

For $m > 1$, assume without loss of generality that $\ell_1 \leq \cdots \leq \ell_{m-1} \leq \ell_m$. We will first try to build a code with $(m-1)$ code words, of lengths $\ell_1, \ldots, \ell_{m-2}, (\ell_{m-1} - 1)$. In order to be able to invoke the induction hypothesis, we do need to check that

$$\left(\sum_{i=1}^{m-2} 2^{-\ell_i}\right) + 2^{-(\ell_{m-1}-1)} \leq 1. \tag{2.2}$$

This can be seen by first noting that

$$1 \;>\; 1 - 2^{-\ell_m} \;\geq\; \sum_{i=1}^{m-1} 2^{-\ell_i} \;=\; \sum_{i=1}^{m} \frac{2^{\ell_{m-1}-\ell_i}}{2^{\ell_{m-1}}} \;=\; \frac{\sum_{i=1}^{m-1} 2^{\ell_{m-1}-\ell_i}}{2^{\ell_{m-1}}}. \tag{2.3}$$

Thus, on the right-hand side, we end up with a fraction of the form $\frac{a}{b} < 1$, where a and b are both integers, with $a < b$. In this case it must follow that $a + 1 \leq b$, and hence $\frac{a+1}{b} \leq 1$. So we get

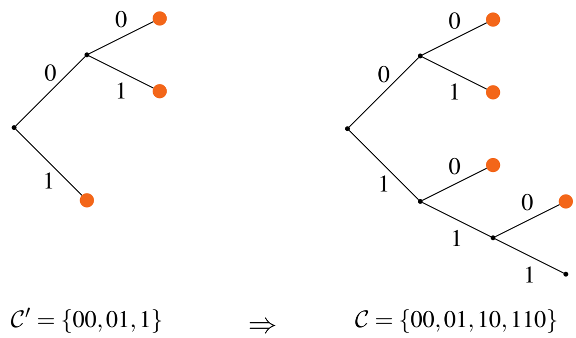
$$1 \;\geq\; \frac{\left(\sum_{i=1}^{m-1} 2^{\ell_{m-1}-\ell_i}\right) + 1}{2^{\ell_{m-1}}} \;=\; \left(\sum_{i=1}^{m-1} 2^{-\ell_i}\right) + 2^{-\ell_{m-1}}$$

$$= \left(\sum_{i=1}^{m-2} 2^{-\ell_i}\right) + 2 \cdot 2^{-\ell_{m-1}}$$

$$= \left(\sum_{i=1}^{m-2} 2^{-\ell_i}\right) + 2^{-(\ell_{m-1}-1)}, \tag{2.4}$$

as desired. So we invoke the induction hypothesis to create a prefix-free code $\mathcal{C}' = \{c'_1, \ldots, c'_{m-1}\}$ with lengths $\ell_1, \ldots, \ell_{m-2}, (\ell_{m-1} - 1)$. The new code $\mathcal{C}$ is then constructed by setting $c_i = c'_i$ for all $i \leq m-2$, and furthermore setting $c_{m-1} = c'_{m-1}|0$ and $c_m = c'_{m-1}|10\cdots0$, padding with enough zeroes to achieve $\ell(c_m) = \ell_m$. This new code is necessarily also prefix-free.

The following image illustrates the induction step for $\ell_1 = \ell_2 = \ell_3 = 2$ and $\ell_4 = 3$. The code is constructed from a prefix-free code $\mathcal{C}'$ with code lengths 2, 2 and 1 by replacing the bottom codeword (of length 1) with two new codewords (of lengths 2 and 3).



■

A stronger version of Kraft's inequality holds as well, this time for uniquely decodable codes:

Theorem 2.2.2 — McMillan inequality. For a uniquely decodable code with image $\mathcal{C} = \{c_1, \ldots, c_m\}$ and codeword lengths $\ell_i := \ell(c_i)$, it holds that

$$\sum_{i=1}^{m} 2^{-\ell_i} \leq 1.$$

Proof. Let $\mathcal{C}$ be a uniquely decodable code as in the theorem statement. We can write

$$S := \sum_{c \in \mathcal{C}} \frac{1}{2^{\ell(c)}} = \sum_{\ell=L_{\min}}^{L_{\max}} \frac{n_\ell}{2^\ell} \tag{2.5}$$

where $L_{\min} = \min_{c \in \mathcal{C}} \ell(c)$, $L_{\max} = \max_{c \in \mathcal{C}} \ell(c)$, and $n_\ell = |\{c \in \mathcal{C} \,|\, \ell(c) = \ell\}|$. Furthermore, for any $k \in \mathbb{N}$, consider the kth power of S,

$$S^k = \sum_{c_1, \ldots, c_k \in \mathcal{C}^k} \frac{1}{2^{\ell(c_1)+\ldots+\ell(c_k)}} = \sum_{\ell=kL_{\min}}^{kL_{\max}} \frac{n_\ell^{(k)}}{2^\ell} \tag{2.6}$$

where $n_\ell^{(k)}$ is defined as $n_\ell^{(k)} = \left|\left\{(c_1, \ldots, c_k) \in \mathcal{C}^k \,\middle|\, \sum_i \ell(c_i) = \ell(c_1|\ldots|c_k) = \ell\right\}\right|$. Note that

$$n_\ell^{(k)} = \sum_{x \in \{0,1\}^\ell} \left|\left\{(c_1, \ldots, c_k) \in \mathcal{C}^k \,\middle|\, c_1|\ldots|c_k = x\right\}\right| \leq \sum_{x \in \{0,1\}^\ell} 1 = 2^\ell \tag{2.7}$$

where the inequality follows from the unique decodability of $\mathcal{C}$. Thus, we can conclude that

$$S^k \leq (L_{\max} - L_{\min}) \cdot k \tag{2.8}$$

for all $k \in \mathbb{N}$, so S^k grows at most linearly in k, from which follows that $S \leq 1$ (for if not, S^k would grow exponentially in k). ■

Kraft's and McMillan's inequality together lead to the conclusion that the lengths of an optimal prefix-free code and an optimal uniquely decodable code coincide:

Corollary 2.2.3 Let P_X be a source. For every uniquely decodable code C, there exists a prefix-free code C' such that $\ell_C(P_X) = \ell_{C'}(P_X)$. Hence,

$$\ell_{\min}^{\text{p.f.}}(P_X) = \ell_{\min}^{\text{u.d.}}(P_X).$$

From now on, we will just write $\ell_{\min}(P_X)$ to denote either of these measures for average length. A code C for which $\ell_C(P_X) = \ell_{\min}(P_X)$ is called **optimal** for the source P_X.

2.3 Shannon's Source-Coding Theorem

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

Theorem 2.3.1 — Shannon's source-coding theorem (for symbol codes). For any source P_X, we have the following bounds:

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

Proof. The proof relies on Kraft's inequality (Section 2.2). Let C be a code, and write ℓ_x for $\ell(C(x))$ as a notational convenience. For the lower bound, we have that

$$\begin{aligned}
H(X) - \ell_C(P_X) &= -\sum_{x \in \mathcal{X}} P_X(x) \log(P_X(x)) - \sum_{x \in X} P_X(x)\ell_x \\
&= \sum_{x \in \mathcal{X}} P_X(x) \left(-\log(P_X(x)) - \log\left(2^{\ell_x}\right)\right) \\
&= \sum_{x \in \mathcal{X}} P_X(x) \log\left(\frac{1}{P_X(x) \cdot 2^{\ell_x}}\right) \\
&\leq \log\left(\sum_{x \in \mathcal{X}} \frac{1}{2^{\ell_x}}\right) && \text{(by Jensen's inequality)} \\
&\leq \log(1) = 0 && \text{(by Kraft's inequality)}
\end{aligned} \tag{2.9}$$

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

$$\ell_x := \left\lceil \log \frac{1}{P_X(x)} \right\rceil, \tag{2.10}$$

and note that

$$\sum_{x \in \mathcal{X}} 2^{-\ell_x} \leq \sum_{x \in \mathcal{X}} 2^{-\log \frac{1}{P_X(x)}} = \sum_{x \in \mathcal{X}} P_X(x) = 1. \tag{2.11}$$

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

$$\begin{aligned}
\ell_C(P_X) &= \sum_{x\in\mathcal{X}} P_X(x)\ell_x \\
&\leq \sum_{x\in\mathcal{X}} P_X(x)\left(\log\frac{1}{P_X(x)}+1\right) \\
&= -\sum_{x\in\mathcal{X}} P_X(x)\log P_X(x) + \sum_{x\in\mathcal{X}} P_X(x) \\
&= H(X)+1. \qquad (2.12)
\end{aligned}$$

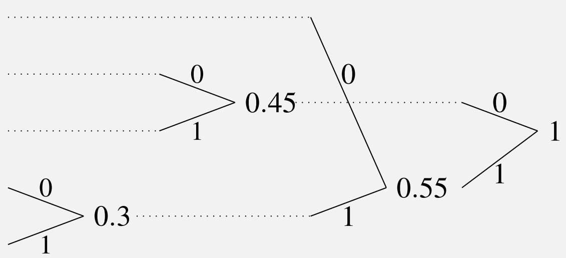
We have thus constructed a code C with $\ell_C(P_X) \leq H(X)+1$, so $\ell_{\min}(P_X) \leq H(X)+1$. ■

2.4 Huffman Codes

Shannon's source-coding theorem shows us that in theory, the minimal code length for a source P_X is roughly $H(X)$. In this section we will investigate **Huffman codes**, which provide an explicit and neat construction for optimal prefix-free codes. A binary Huffman code for a source P_X is constructed by iteratively pairing the two symbols with the smallest probability together, building a binary tree on the way. This is best explained by example:

Example 2.4.1 — Binary Huffman code. Let the random variable X be given with $\mathcal{X} = \{a,b,c,d,e\}$ and $P_X(a) = P_X(b) = 0.25$, $P_X(c) = 0.2$, and $P_X(d) = P_X(e) = 0.15$. The following is a binary Huffman code for P_X:

x	$P_X(x)$	code
a	0.25	10
b	0.25	00
c	0.2	01
d	0.15	110
e	0.15	111



We build up the tree from left to right, pairing the symbols (or groups of symbols) with smallest (combined) probabilities at every step. The codeword for every symbol is then determined by following the branches of the tree *from right to left* until the symbol is reached. Note that this way, the symbols with the smallest probabilities get assigned the longest codewords (paths).

The average codeword length for this code is

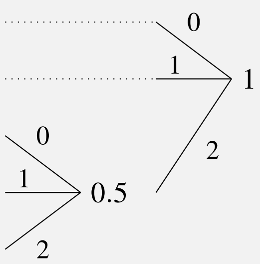
$$0.25\cdot 2 + 0.25\cdot 2 + 0.2\cdot 2 + 0.15\cdot 3 + 0.15\cdot 3 = 2.3. \qquad (2.13)$$

This very close to the entropy $H(X) \approx 2.285$. The average codeword length lies between $H(X)$ and $H(X)+1$.

The above was an example of how to construct *binary* Huffman codes. We can also generate Huffman codes for larger alphabets, resulting in ternary, quaternary, or, more generally, **d-ary Huffman codes**.

Example 2.4.2 — Ternary Huffman code. We build a Huffman code with the alphabet $\{0,1,2\}$ for the same distribution as in Example 2.4.1.

x	$P_X(x)$	code
a	0.25	0
b	0.25	1
c	0.2	20
d	0.15	21
e	0.15	22



Exercise 2.4.1 Use the above procedure to construct a ternary code for the source P_X with $\mathcal{X} = \{a,b,c,d,e,f\}$ and $P_X(a) = P_X(b) = 0.25$, $P_X(c) = 0.2$, $P_X(d) = P_X(e) = P_X(f) = 0.1$. Can you find another code with a smaller average codeword length? ■

We have to be careful, because with an alphabet size of greater than 2, the above procedure does not always give an optimal code! In fact, a d-ary code is only optimal if $|\mathcal{X}|$ is of the form $k(d-1)+1$ for some $k \in \mathbb{N}$. This ensures that at every step, we can combine exactly d symbols to use the alphabet at full capacity. The ternary code in Example 2.4.2 is optimal because $|\mathcal{X}| = 5 = 2(3-1)+1$, but the code you constructed in Exercise 2.4.1 is not. To remedy this, one can add one or more 'dummy' symbols to the source (each with probability zero) until an appropriate size of $\mathcal{X}$ of the form $|\mathcal{X}| = k(d-1)+1$ for some $k \in \mathbb{N}$ is reached. The codewords for those dummy symbols are discarded at the end.

We now show that Huffman codes indeed have optimal code length.

Theorem 2.4.1 — Optimality of Huffman codes. Let P_X be a source, and let C^* be the associated Huffman code. For any other uniquely decodable code C' with the same alphabet,

$$\ell_{C^*}(P_X) \leq \ell_{C'}(P_X).$$

Proof (sketch). We sketch the proof for binary codes. The proof idea extends to any alphabet size.

Let $a,b \in \mathcal{X}$ be the two symbols with lowest probability, and $P_X(a) \leq P_X(b)$. Suppose, for some prefix-free code C', that $\ell_a < \ell_b$. We can be in one of two cases:

1. ℓ_b is not the maximal codeword length. In other words, some c exists with $P_X(c) \geq P_X(a)$ and $\ell_c > \ell_a$.
2. ℓ_b is the maximal codeword length. Then there exists a 'sibling' c in the code tree such that $\ell_c = \ell_b$ (if not, we can remove the last bits of the code for b by prefix-freeness). For this c, $P_X(c) \geq P_X(a)$ and $\ell_c > \ell_a$.

In both cases, we can swap the codewords for a and c, achieving a code that is at least as efficient as C'. Therefore, we may assume that for all source symbols $x_i, x_j \in \mathcal{X}$ with $P_X(x_i) \leq P_X(x_j)$, $\ell_{x_i} \geq \ell_{x_j}$.

Using this assumption, we can transform any code C' into a Huffman code using induction on the size of the source. The full proof can be found in [CT] (Theorem 5.8.1). [**Chris:** I'm not sure if this "proof" is useful in its current form, I always struggled to give some good intuition here instead of the painfully long proof in CT. I think that the short proof in the solution of Ex 5.16 in MacKay on page 105 is also hard to follow...] ■

Example 2.4.3 — 20 questions. In the game of '20 questions' (20Q) the goal is to identify an object from a set of objects using (at most) 20 yes/no questions. Please go and play it online right now if you have never heard of it before. Assume we know the probability distribution P_X over all possible objects, what is the most efficient sequence of questions to ask in order to determine the object? We can use Huffman coding to answer this question!

On the one hand, the Huffman code for P_X has optimal average code length and we could (in principle) ask the player about the first, second, third, etc. bit of the object in question (admittedly, these questions would be rather boring...). On the other hand, we can see the sequence of (questions and) answers as the 'code' for an object, because every object as a unique sequence of yes/no answers. The number of questions asked is the length of the codeword. By Shannon's source-coding theorem,

$$H(X) \leq \text{expected number of questions} \leq H(X)+1,$$

and the Huffman code procedure can be used to determine the optimal sequence of questions.

As we have seen, the average codeword length of Huffman codes is theoretically optimal. However, Huffman codes (and symbol codes in general) still have a number of disadvantages:

- When compressing, for example, an English text symbol-by-symbol, the probability distribution for each position may depend on the string of text that precedes it: for example, the letter `n` is a lot more likely than the letter `a` if it comes after the string `informatio`. Given this change of distribution, the Huffman code may not produce the shortest possible code. This can be resolved by recomputing the Huffman code after every symbol, but this results in a lot of overhead.
- The average codeword length is upper bounded by $H(X)+1$. This additive cost of 1 bit is fine when $H(X)$ is very large, but can be a significant overhead when $H(X)$ is small itself.

2.5 Arithmetic Codes

In this section, we study a different kind of code that can handle context-dependent distributions, unlike the Huffman code.

Definition 2.5.1 — Standard binary representation. The standard binary representation of a real number $r \in [0,1)$ is a (possibly infinite) string of bits $c_1c_2\cdots$ such that

$$r = \sum_i c_i \cdot 2^{-i},$$

where by convention, 0 is represented by the string 0.

Not all reals in $[0,1)$ have a finite representation, but any interval $[a,b)$ with $0 \leq a < b \leq 1$ contains at least one number with a finite binary representation.

Example 2.5.1 The following table lists some numbers $r \in [0,1)$ and their standard binary representation.

r	binary representation of r
$1/2$	1
$1/3$	$01010101\cdots$
$1/4$	01
$3/4$	11
$13/16$	1101
$13/32$	01101

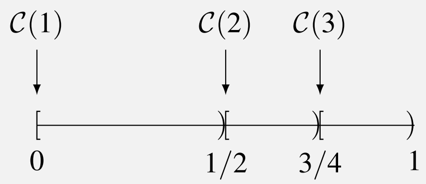
Note that 1101 is also the binary form of the natural number 13. Adding a 0 on the left divides the represented value by 2.

These binary representations of numbers in the interval $[0,1)$ give rise to a very elegant code:

Definition 2.5.2 — Arithmetic code. Given a source P_X with $\mathcal{X} = \{x_1,..,x_m\}$, construct the arithmetic code as follows. Divide the interval $[0,1)$ into disjoint subintervals $I_{x_j} = [a_j, a_{j+1})$, where $a_1,\ldots,a_{m+1}$ are defined such that $a_{j+1} - a_j = P_X(x_j)$, and $a_1 = 0, a_{m+1} = 1$.

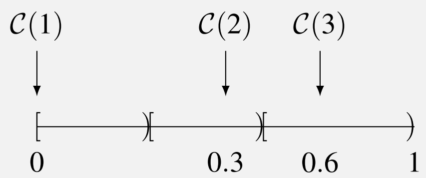
The encoding $AC(x_j)$ of the element x_j is the (shortest possible) standard binary representation of some number in the interval I_{x_j}.

Example 2.5.2 Let X be a random variable with $\mathcal{X} = \{1,2,3\}$ and $P_X(1) = \frac{1}{2}, P_X(2) = P_X(3) = \frac{1}{4}$. The arithmetic code is constructed by first determining the intervals:



This image results in the arithmetic code $\mathcal{C}$ with $\mathcal{C}(1) = 0$ (the representation of 0), $\mathcal{C}(2) = 1$ (the representation of $\frac{1}{2}$), $\mathcal{C}(2) = 11$ (the representation of $\frac{3}{4}$).

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



Proposition 2.5.1 For any (X, P_X), the arithmetic code has average length $\ell_{AC}(P_X) \leq H(X) + 1$.

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

$$2^{-\ell_x} = 2^{-\lceil \log(1/P_X(x)) \rceil} \leq 2^{-\log(1/P_X(x))} = 2^{\log P_X(x)} = P_X(x). \tag{2.14}$$

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

Repeating this argument for every x, we obtain

$$\ell_{AC}(P_X) = \mathbb{E}[\ell(AC(X))] = \sum_x P_X(x)\ell(AC(x)) \leq \sum_x P_X(x)(-\log P_X(x)+1) = H(X)+1. \tag{2.15}$$

■

As we can see from Example 2.5.2, this construction for arithmetic codes does not necessarily yield prefix-free codes. However, at the expense of one extra bit of code (on average), the construction can be adapted into a prefix-free code. One example of this is the **Shannon-Fano-Elias code** (see [CT], Section 5.9 or wikipedia), which provides a more sophisticated way of selecting a number within each interval than simply selecting the number with the shortest binary representation. This alternative selection procedure ensures prefix-freeness. Another option is to select *binary intervals* within each interval:

Definition 2.5.3 — Binary interval. A binary interval is an interval of the form

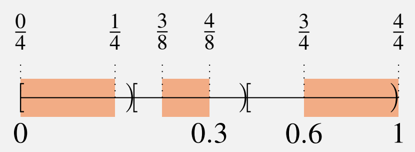
$$\left[\frac{s}{2^\ell}, \frac{s+1}{2^\ell}\right)$$

with $s, \ell \in \mathbb{N}$ and $0 \leq s < 2^\ell$. The **name** of the interval is the binary representation of s (as a natural number) padded with zeroes on the left to reach length ℓ.

Definition 2.5.4 — Arithmetic code (prefix-free version). The prefix-free arithmetic code is identical to Definition 2.5.2, except that the encoding $AC^{pf}(x_j)$ of the element x_j is now the name of the largest binary interval that fits entirely in I_{x_j}.

Similarly to Proposition 2.5.1, it can be shown that for any source P_X, $\ell_{AC^{pf}}(P_X) \leq H(X)+2$. Note that we get prefix-freeness only at the expense of an extra bit on average.

Example 2.5.3 Let Y be the random variable as in Example 2.5.2, that is, $P_Y(1) = P_Y(2) = 0.3$ and $P_Y(3) = 0.4$. The prefix-free code for Y is constructed as follows:



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

The arithmetic code is slightly less efficient than the Huffman code in terms of average codeword length. A big advantage is the way it is able to adapt to changing distributions, such as when we are encoding a stream of English text. Suppose we are given the (not necessarily i.i.d.) random variables $X_1, X_2, \ldots, X_n$, and we want to encode the source $P_{X_1X_2\cdots X_n}$. We start by dividing the interval $[0,1)$ into subintervals according to P_{X_1}. If, for example, the event $X_1 = b$ happens, we zoom into the interval corresponding to b, and subdivide *that* interval according to $P_{X_2|X_1}$, so that the sizes of these intervals add up to $P_{X_1}(b)$. The concept of arithmetic coding is exploited as an accessibility tool in the keyboard alternative Dasher, invented by the group of David MacKay at Cambridge University, UK.

2.6 Asymptotic Equipartition Property (AEP)

In this section, we consider the possibility of encoding blocks of symbols, rather than just one symbol at at time. We restrict our attention to sources that are *real* random variables. The following definition of converging random variables may remind you of a converging sequence of numbers. Recall that a sequence $x_1, x_2, x_3, \dots$ of numbers converges to x if $\forall \varepsilon > 0\ \exists n_0\ \forall (n \geq n_0) : |x_n - x| < \varepsilon$. We denote this by writing $x_n \xrightarrow{n\to\infty} x$.

Definition 2.6.1 — Converging random variables. A sequence $X_1, X_2, X_3, \dots$ of real random variables converges to a random variable X, if it satisfies one of the following definitions:

in probability	(notation $X_n \xrightarrow{p} X$)	if $\forall \varepsilon > 0$, $P[|X_n - X| > \varepsilon] \xrightarrow{n\to\infty} 0$
in mean square	(notation $X_n \xrightarrow{m.s.} X$)	if $\mathbb{E}[(X_n - X)^2] \xrightarrow{n\to\infty} 0$
almost surely	(notation $X_n \xrightarrow{a.s.} X$)	if $P[\lim_{n\to\infty} X_n = X|] = 1$

where the definition of $X_n \xrightarrow{a.s.} X$ can be interpreted as $P[\{\omega \in \Omega \mid X_n(\omega) \xrightarrow{n\to\infty} X(\omega)\}] = 1$.

[**Yfke:** maybe add example here?] [**Chris:** yes, or at least refer to wikipedia]
In general, the following implications hold (although their converses do not):

$$X_n \xrightarrow{m.s.} X \quad \Rightarrow \quad X_n \xrightarrow{p} X \tag{2.16}$$

$$X_n \xrightarrow{a.s.} X \quad \Rightarrow \quad X_n \xrightarrow{p} X \tag{2.17}$$

The following important weak law of large numbers states that if we sample several times from the same distribution, the average converges (in probability) to the expected value of the distribution.

Theorem 2.6.1 — Weak Law of Large Numbers. Let $X_1, X_2, \dots$ be real i.i.d. random variables with mean $\mu = \mathbb{E}[X_i]$ and variance $\sigma^2 = \mathbb{E}[(X_i - \mu)^2] < \infty$. Define the random variables

$$S_n := \frac{1}{n} \sum_{i=1}^{n} X_i.$$

Then $S_n \xrightarrow{p} \mu$.

This important law has an entropy variant, which follows almost directly:

Theorem 2.6.2 — Asymptotic Equipartition Property (AEP). Let $X_1, X_2, X_3, \dots$ be real i.i.d. random variables with distribution P_X. Then

$$-\frac{1}{n} \log P_{X_1 \cdots X_n}(X_1, \dots, X_n) \xrightarrow{p} H(X).$$

(Note that $P_{X_1 \cdots X_n}(X_1, \dots, X_n)$ is itself a random variable, and $H(X)$ can be regarded as a constant random variable.)

Proof. Since the variables X_i are independent, so are the random variables $\log P_X(X_i)$. Then

$$\begin{aligned} -\frac{1}{n} \log P_{X_1 \cdots X_n}(X_1, \dots, X_n) &= -\frac{1}{n} \sum_{i=1}^{n} \log P_X(x_i) \\ &\xrightarrow{p} -\mathbb{E}[\log P_X(X_i)] = H(X) \end{aligned} \tag{2.18}$$

by the weak law of large numbers. ∎

Example 2.6.1 — Biased coin flip. Consider flipping a biased coin, with probability of 1 (heads) being $p_1 = 0.1$ and probability of 0 (tails) being $p_0 = 0.9$. (This is an example of a **Bernoulli distribution** with success probability p_1 and failure probability $p_0 = 1 - p_1$.) We can compute that $H(X) = h(0.1) \approx 0.469$.

Clearly, for every n, $P_{X_1 \cdots X_n}(x_1 \cdots x_n) = p_0^{n-r(x)} p_1^{r(x)}$, where $r(x)$ is the number of ones in the string $x = x_1 \cdots x_n$. In particular, this gives us the **binomial distribution** by considering the random variable $R = r(X_1, \ldots, X_n)$, since $P(r) = \binom{n}{r} p_0^{n-r} p_1^r$.

[**Yfke:** the connection with the AEP is not really clear in this example] [**Chris:** it only becomes clear from the slides! There, one can see the concentration, maybe we can actually re-use MacKay's figures here.] [**Chris:** also, we introduced these distributions now in Chapter 1 already.]

Definition 2.6.2 — Typical set. The typical set $A_\varepsilon^{(n)}$ with respect to P_X is the set of strings $(x_1, \ldots, x_n) \in \mathcal{X}^n$ such that

$$2^{-n(H(X)+\varepsilon)} \leq P_{X^n}(x_1, \ldots, x_n) \leq 2^{-n(H(X)-\varepsilon)}.$$

The typical set is relatively small, but contains almost all of the probability mass. We start by establishing some general properties of typical sets.

Proposition 2.6.3 A typical set $A_\varepsilon^{(n)}$ satisfies the following:

1. For all $(x_1, \ldots, x_n) \in A_\varepsilon^{(n)}$,
$$H(X) - \varepsilon \leq -\frac{1}{n} \log P_{X^n}(x_1, \ldots, x_n) \leq H(X) + \varepsilon.$$
2. $P[A_\varepsilon^{(n)}] > 1 - \varepsilon$ (for large enough n).
3. $|A_\varepsilon^{(n)}| \leq 2^{n(H(X)+\varepsilon)}$.
4. $|A_\varepsilon^{(n)}| \geq (1-\varepsilon) 2^{n(H(X)-\varepsilon)}$ (for large enough n).

Proof.

1. This is immediate from the definition (take the logarithm and divide by $-n$).
2. This follows from the Asymptotic Equipartition Property: for all $\varepsilon > 0$, $P[|-\frac{1}{n}\log P_{X^n}(X_1, \ldots, X_n) - H(X)| > \varepsilon] \xrightarrow{n\to\infty} 0$, that is,
$$\forall(\varepsilon > 0)\ \forall(\delta > 0)\ \exists n_0\ \forall(n \geq n_0)\ P[|-\frac{1}{n}\log P_{X^n}(X_1, \ldots, X_n) - H(X)| \leq \varepsilon] > 1 - \delta. \tag{2.19}$$
By choosing $\delta := \varepsilon$, the result follows from the first property.
3. First, observe that
$$1 = \sum_{\vec{x} \in \mathcal{X}^n} P_{X^n}(\vec{x}) \geq \sum_{\vec{x} \in A_\varepsilon^{(n)}} P_{X^n}(\vec{x}) \geq |A_\varepsilon^{(n)}| \cdot 2^{-n(H(X)+\varepsilon}, \tag{2.20}$$
where the last inequality follows by property 1. The result now follows by multiplying both sides of the equation by $2^{n(H(X)+\varepsilon)}$.
4. By property 2, we can choose an n large enough so that
$$1 - \varepsilon < P[A_\varepsilon^{(n)}] = \sum_{\vec{x} \in A_\varepsilon^{(n)}} P_{X^n}(\vec{x}) \leq |A_\varepsilon^{(n)}| \cdot 2^{-n(H(X)-\varepsilon)}, \tag{2.21}$$
where again, the last inequality follows by property 1.

■

Typical sets and their properties allow us to code a source P_X, n symbols at a time, in either a lossy or a lossless way. For a lossy code, simply assign binary labels of length (at most) $\lceil n(H(X)+\varepsilon)\rceil$ to the elements of $A_\varepsilon^{(n)}$, and assign some constant (dummy) codeword to all elements outside of the set. Decoding this dummy codeword will result in an error (data loss), but this error occurs with probability at most ε.

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

Theorem 2.6.4 Let $X_1,\ldots,X_n$ be i.i.d. real random variables with respect to the set $\mathcal{X}$, and distributed according to P_X. Let $\varepsilon > 0$. Then there exists a lossless code $\mathcal{X}^n \to \{0,1\}^*$ such that, for sufficiently large n, $\mathbb{E}[\frac{1}{n}\ell(X^n)] \leq H(X)+\varepsilon$.

Proof. Consider the code described above: the code consist of a flag bit (indicating whether or not the element is inside the typical set), followed by either a short label (for elements in the typical set) or a longer one (for elements outside of it).

Let $\varepsilon' > 0$ (we will specify the value of ε' later). Let n be large enough such that $P[A_{\varepsilon'}^{(n)}] > 1-\varepsilon'$ (see Proposition 2.6.3). Then

$$\begin{aligned}
\mathbb{E}[\ell(X^n)] &= \sum_{\vec{x}\in\mathcal{X}^n} P_{X^n}(\vec{x})\ell(\vec{x}) \\
&= \sum_{\vec{x}\in A_{\varepsilon'}^{(n)}} P_{X^n}(\vec{x})\ell(\vec{x}) + \sum_{\vec{x}\notin A_{\varepsilon'}^{(n)}} P_{X^n}(\vec{x})\ell(\vec{x}) \\
&\leq P[A_{\varepsilon'}^{(n)}]\cdot(\lceil n(H(X)+\varepsilon')\rceil+1) + P[\overline{A_{\varepsilon'}^{(n)}}]\cdot(\lceil n\log|\mathcal{X}|\rceil+1) \\
&\leq P[A_{\varepsilon'}^{(n)}]\cdot(n(H(X)+\varepsilon')+2) + P[\overline{A_{\varepsilon'}^{(n)}}]\cdot(n\log|\mathcal{X}|+2) \\
&\leq n(H(X)+\varepsilon') + \varepsilon'\cdot n\log|\mathcal{X}|+2 \\
&= n(H(X)+\varepsilon'),
\end{aligned} \tag{2.22}$$

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

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

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

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

Theorem 2.6.5 Let $X_1, \ldots, X_n$ be i.i.d. random variables distributed according to P_X. For any $\delta < \frac{1}{2}$, and any $\delta' > 0$, if $P[B_\delta^{(n)}] > 1 - \delta$, then

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

for sufficiently large n.

Proof. Let $\delta, \varepsilon < \frac{1}{2}$, and consider some $B_\delta^{(n)}$ such that $P[B_\delta^{(n)}] > 1 - \delta$. We know that by Proposition 2.6.3, $P[A_\varepsilon^{(n)}] > 1 - \varepsilon$, for large enough n. Thus, by the union bound,

$$\begin{aligned} 1 - \varepsilon - \delta < 1 - P[\overline{A_\varepsilon^{(n)}}] - P[\overline{B_\delta^{(n)}}] \qquad & \leq P[A_\varepsilon^{(n)} \cap B_\delta^{(n)}] \\ &= \sum_{\vec{x} \in A_\varepsilon^{(n)} \cap B_\delta^{(n)}} P_{X^n}(\vec{x}) \\ &\leq \sum \vec{x} \in A_\varepsilon^{(n)} \cap typsetb 2^{-n(H(X) - \varepsilon} \\ &= |A_\varepsilon^{(n)} \cap B_\delta^{(n)}| \cdot 2^{-n}(H(X) - \varepsilon) \\ &\leq |B_\delta^{(n)}| \cdot 2^{-n}(H(X) - \varepsilon). \end{aligned} \tag{2.23}$$

Rearranging this expression and taking the logarithm, we get

$$H(X) - \varepsilon + \frac{1}{n} \log(1 - \varepsilon - \delta) < \frac{1}{n} \log |B_\delta^{(n)}|. \tag{2.24}$$

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

$$H(X) - \delta' < \frac{1}{n} \log |B_\delta^{(n)}|, \tag{2.25}$$

as desired. [**Yfke:** I'm not sure what happens to δ' here: the proof should work for any $\delta' > 0$, aren't we restricting the freedom here? Or can we always find an epsilon to accommodate for the values of δ and δ' that are given?] ∎

2.7 Comments

1. we can add an exercise / note on the role of relative entropy in coding theory (the added inefficiency if we use an optimal code for some distribution Q to encode symbols from P, resulting in an average code length of $H(P_X) + D(P||Q)$)