AICC II
Prof. Bixio Rimoldi

Benjamin Bovey

Semester of Spring 2019

19th February 2019

As opposed to the first AICC course, where we were mostly presented with tools, we will now see more applications of these tools for communication and computation. Mainly, we will see 3 applications in the first part of the semester:

- **Source coding** (compressing information)
- **Cryptography** (authentication / privacy / integrity of information)
- **Channel Coding** (dealing with noise and loss of information / protecting the information from natural damages)

What these three have in common is the idea of storing and communicating information. The notion of entropy, which will come up quite often, will also be important.

Basic probability review

Special case first: finite sample space Ω and uniform distribution. $\Omega = \{\omega_1, \omega_2, \dots, \omega_n\}$. Events: $E \subseteq \Omega$. Then:

$$P(E) = \frac{|E|}{|\Omega|} \qquad \text{(uniform distribution)} \tag{1}$$

Definition 1 (conditional probability)**.** Let E, F be two events. Then, the probability that event E occurs knowing that F has occured:

$$P(E|F) = \frac{|E \cap F|}{|F|} \tag{2}$$

Intuitively, you restrict the sample space to F only, because you *know* that F has happened: this translates to the division by the cardinality of F instead of the cardinality of Ω. The intersection of E and F follows from the fact that the sample space is restricted to F: if there exists elements that are in E but not in F, they are outside of the new sample space F, which means that these elements CANNOT occur in conjunction with F. Therefore, we take the intersection of E and F to assure that these elements are not taken into account in the computation.

Theorem 2 (Law of total probability)**.** *Let E and F be two events in Ω, and let F^C denote the complement of F. Then:*

$$P(E) = P(E|F)P(F) + P(E|F^C)P(F^C) \tag{3}$$

This follows quite directly from the fact that $E = (E \cap F) \cup (E \cap F^C)$, so $P(E) = P(E \cap F) + P(E \cap F^C)$ ...

Remark (divide and conquer)*.* You can sometimes create a partition of your sample space, in a way that allows you to better apply the numbers you are given (p.27-28). This method is called *divide and conquer.*

Theorem 3 (Bayes)**.** *Bayes' theorem allows you to compute $p(F|E)$, given that you know $p(E|F), p(E)$ and $p(F)$:*

$$p(F|E) = \frac{p(E|F)p(F)}{p(E)} \tag{4}$$

Remark (application)*.* This is useful, in real scenarios, when either one of $p(E|F)$ and $p(F|E)$ is easily observable, but the other isn't. For example, policemen may observe how many people are driving drunk knowing that they have had an accident (they just test the driver after the accident), but they cannot observe how many people are having an accident knowing that they are driving drunk (they can not really test drivers, and then let them drive drunk just to check if they have an accident or not).

Definition 4. A **random variable** X is a function $X : \Omega \to \mathbb{R}$. It is attached a *probability distribution function* $p_X(x)$, which represents the probability that X will take on the value x, that is, that the following event E occurs:

$$E = \{\omega \in \Omega : X(\omega) = x\} \tag{5}$$

Hence,

$$p_X(x) = p(E) = \sum_{\omega \in E} p(\omega) \tag{6}$$

The set of all possible values of X is sometimes called the *alphabet* of X, written with more curly letters like $\mathcal{A}$.

Theorem 5 (two random variables)**.** *Let $X : \Omega \to \mathbb{R}$ and $Y : \Omega \to \mathbb{R}$ be two random variables. The probability of the event $E = \{X = x\} \cap \{Y = y\} (= \{X = x \land Y = y\})$ is*

$$p_{X,Y}(x, y) = \sum_{\omega \in E} p(\omega) \tag{7}$$

We can compute p_X (or p_Y, similarly) from $p_{X,Y}$:

$$p_X(x) = \sum_{y} p_{X,Y}(x, y) \tag{8}$$

In one sense, we "fix in place" the value of x and "scroll through" all possible values of y, and add their probabilities up. Here, p_X is called the **marginal distribution** of $p_{X,Y}(x, y)$ with respect to x.

21st February 2019

Definition 6 (expected value)**.** The **expected value**, or **mean** of a random variable $X : \Omega \to \mathbb{R}$, can be computed as

$$E[X] = \sum_{x \in \mathcal{A}(X)} x p_X(x) \quad \text{(requires } p_X\text{)}, \tag{9}$$

or as

$$E[X] = \sum_{\omega \in \Omega} X(\Omega) p(\omega). \tag{10}$$

One could say that the first way is "calculating over the codomain", and the second way is "calculating over the domain" (of X).

Remark. The expected value is a linear operation. Let $X_1, X_2, \dots, X_n$ be random variables from Ω to $\mathbb{R}$, and let $\lambda_1, \lambda_2, \dots, \lambda_n$ be real numbers. Then

$$E \underbrace{\left[\sum_{i=1}^{n} X_i \lambda_i\right]}_{\text{random variable}} = \sum_{i=1}^{n} \lambda_i E[X_i] \tag{11}$$

Extending notions from events to random variables

The notion of independent events extends to random variables. Recall that two events E and F are independent iff $p(E|F) = p(E)$, which is equivalent to saying that $p(E \cap F) = p(E)p(F)$.

Similarly, two random variables may be independent if the value taken by one does not influence the value taken by the other.

Definition 7 (independent random variables)**.** We say that two random variables $X, Y : \Omega \to \mathbb{R}$ are **independent** iff

$$p_{X,Y}(x, y) = p_X(x) p_Y(y) \tag{12}$$

More generally, n random variables are independent iff

$$p_{S_1,\dots,S_n} = \prod_{i=1}^{n} p_{S_i} \tag{13}$$

From there, we can also extend the notion of conditional probability to random variables.

Definition 8. We define the **conditional probability of two random variables** $X, Y : \Omega \to \mathbb{R}$ as

$$p(X = x|Y = y) = \frac{p(X = x \land Y = y)}{p(Y = y)}. \tag{14}$$

Remark. The following statements are equivalent:

$$p_{X,Y}(x, y) = p_X(x) \tag{15}$$

$$p_{X|Y}(x|y) = p_X(x) \tag{16}$$

$$p_{Y|X}(y|x) = p_Y(y) \tag{17}$$

Remark (useful trick). If you are asked to check the independence of X and Y, you don't have to check the equality of (16) or (17). You just have to find the expression for the left-hand side function, and see if it depends on the other variable ($\implies$ they are NOT independent), or if it is just a function of one variable ($\implies$ they are independent).

Theorem 9 (consequence of the independence of random variables)**.** *In all cases, $E[X+Y] \neq E[X]+E[Y]$. However, if X and Y are* independent*, we also have that*

$$E[XY] = E[X]E[Y] \tag{18}$$

Source & Entropy

The main object that will interest us when studying source coding is the source itself. The question of the definition of a source took mathematicians and computer scientists a while to answer. We can loosely model a source as a black box that outputs *information*: we are not really interested in its inner mechanisms, but rather in what comes out of it, that is, *information.* This information could take many forms, for example sequences of symbols: since we are considering sources from a computer science point of view, we will be interested in sources that shite out sequences of numbers.
The notion of *entropy* comes into the frame when we realize that a symbol that can be *predicted* before it comes out of the source provides no new information. For example, if the sequence of numbers coming out of the source is `1, 1, 5, 5, 3, 3, 19, 19, 1, ...`, we quickly realize that we do not need to store the second number of each pair, as it brings no new information to the table.
An important observation that we can make at this point (it was initially made by Hartley in 1929) is that this link between information and entropy can be modeled very elegantly by random variables! If we think about it, the core idea of a random variable is that it gives you a number that you cannot certainly predict (hence, in fact, the name of *random* variable). If we choose to use this model, a source can be viewed as outputting a sequence of random variables.
Let us now consider a source outputting a sequence of random variables, call them $S_1, S_2, S_3, \ldots, S_n$. Another fundamental question we may ask ourselves is: <u>how much information is actually conveyed by each individual symbol?</u> A partial answer was given by Hartley, that is, that this must depend on the alphabet of the random variable:

Definition 10. The **alphabet** of a random variable is the codomain of the random variable, that is, the set of all values that the random variable may take.

Indeed, the bigger the alphabet, the more information it can carry, as there are more possibilities for each symbol, and therefore less predictability. With basic combinatorics, we can see that there are $|\mathcal{A}|^n$ possible length-n sequences $(s_1, s_2, \ldots, s_n)$. Therefore, the amount of information carried by S_i is $\log|\mathcal{A}|$.

Example 11. Imagine this is the sequence of good days (1) and bad days(0) in London during a year:

$$(s_1, s_2, \ldots, s_{365}) = (0, 1, 1, 0, 1, \ldots, 0, 1, 1).$$

There is a lot of entropy, it is unpredictable. It would therefore be hard to optimize the storage of this information better. San Diego:

$$(\overbrace{0, 0, 0, \ldots, 0}^{24 \text{ zeros}}, 1, \overbrace{0, 0, 0, \ldots, 0, 0}^{340 \text{ zeros}}).$$

This is a very predictable sequence, which would allow us to just store $(24, 1, 340)$ rather than storing all 365 bits.

Entropy redefined by Shannon

Shannon, in 1948, gave a new formula for the amount of information carried by the random variable $S \in \mathcal{A}$. He said that this amount *is* in fact the entropy itself, and gave this formula for the entropy $H(S)$:

$$H(S) = -\sum_{s \in \text{supp}(P_S)} p_S(s) \cdot \log_b\big(p_S(s)\big) \tag{19}$$

Remark. We may observe some things:

- $s \in \text{supp}(S)$ is needed because $\log(0)$ is not defined. However, if we are using the convention $0 \cdot \log(0) = 0$, then we can simplify the notation like this:

$$H(S) = -\sum_{s\in\mathcal{A}} p_S(s)\log_b\big(p_S(s)\big);$$

- when $b = 2$ then the unit is the bit. By default, $H(S) = H_2(S)$;
- we may rewrite the formula as $H(S) = \sum_{s\in\mathcal{A}} p_S(s)\underbrace{\Big(-\log_b\big(p_S(s)\big)\Big)}_{\text{rand. var. } X} = E[X]$, because this is the expression of the expected value of a random variable X.

Example 12. Let the random variable $S \in \mathcal{A}$ be uniformly distributed. Then

$$\begin{aligned} H(S) &= -\sum_{s\in\text{supp}(P_S)} \underbrace{p_S(s)}_{\frac{1}{|\mathcal{A}|}}\log_b\big(\underbrace{p_S(s)}_{\frac{1}{|\mathcal{A}|}}\big) \\ &= \end{aligned}$$

So, Hartley and Shannon agree when the random variable has a uniform distribution.

Entropy extends to any number of random variables.

26th February 2019

We saw last time that a source can mathematically be modeled as one or more random variables, each being described by its probability mass function. We saw that the entropy is a number which represents the ultimate amount of bits (not necessarily, but generally, binary) needed to represent a random variable (and therefore a source).

Example 13. $S \in \{0, 1\} = \mathcal{A}, p_S(0) = P$

$$\begin{aligned} H(S) &= -\sum_{s\in\mathcal{A}} p_S(0)\log_b\big(p_S(0)\big) \\ &= \underbrace{-p\log p - (1-p)\log(1-p)}_{h(p)\text{ function of } p} \end{aligned}$$

IT inequality

Many results in information theory are the consequence of the following inequality.

Theorem 14 (IT inequality)**.** *Let $r > 0$. Then*

$$\log_b(r) \leq (r-1)\log_b(e) \tag{20}$$

with the equality iff $r = 1$.

Theorem 15 (Entropy bounds)**.** *Let $s \in \mathcal{A}$. Then*

$$0 \leq H(s) \leq \log|\mathcal{A}| \tag{21}$$

with the first inequality holding iff $S =$ const., and the second inequality holding iff $p_S(s) = \frac{1}{|\mathcal{A}|}$.

Example 16. Let S be your 4-digit lock number. Then $S = \{0, 1, \ldots, 9999\}$. Let's say you choose your lock number at random: then $H(s) = \log 10^4$. However, if your grandma always chooses 0000, then s is a constant, and $H(S) = 0$. A random (least previsible) choice has the most entropy, and a constant (most previsible) choice has the least entropy (zero). This makes sense, and it also means that the random lock number carries the most information, and the constant one carries the least information.

Let's apply this to sources.

Example 17. Let $S_1, S_2, \dots, S_n$ be a sequence of coin flips. Then $S_i \in \{0, 1\}$, and $P_S(s) = \frac{1}{2}$.
Intuitively, we are flipping n coins, and it should take n bits to describe the result (a length-n bitstring).

$$\underbrace{P_{S_1,S_2,\dots,S_n}(s_1, s_2, \dots, s_n)}_{\text{abbreviate as } P(s_1,\dots,s_n)} = \prod_{i=1}^{n} P(s_i) = \left(\frac{1}{2}\right)^n$$
$$H(S_1, S_2, \dots, S_n) = \log(|\mathcal{A}|^n) = n \log 2 = n$$

PERSONAL NOTE: recall that when we write $P(s_1, s_2, \dots, s_n)$, we mean the probability of getting the sequence $(s_1, s_2, \dots, s_n) \in \mathcal{A}^n$. Then the cardinality of the alphabet is $|\mathcal{A}^n| = |\mathcal{A}|^n$.

Source coding

The setup we have is the following: let's say we have a source emitting $S_i \in \mathcal{A}$ towards an encoder with an *encoding map* (a function $\mathcal{A} \to \mathcal{C}$) called Γ. We're going to map each element of the alphabet into a codeword, for example, each letter of the word `dinner`. For example: d$\to$ 000, i$\to$ 010, ...
The encoder is specified by

- an input alphabet $\mathcal{A}$, which is the alphabet of the source
- an output alphabet $\mathcal{C}$
- the encoding map $\Gamma : \mathcal{A} \to \mathcal{C}$

The code is a set of codewords, which are the output of the Γ map. The Γ map is always one-to-one and onto (bijective), but this doesn't mean that we can necessarily go back from the code words to the words, since we usually concatenate the output.

Example 18. Let $\mathcal{A} = \{\mathtt{H}, \mathtt{E}, \mathtt{L}, \mathtt{O}\}$ and $\mathcal{C} = \{01, 10, 0, 11\}$ (Γ maps them in the written order). Then $\Gamma : \mathcal{A} \to \mathcal{C}$ encodes the word `HELLO` to the bitstring `01100011`. The conversion is easy in this direction, but when trying to decode the message, we run into a difficulty: the bitstring could either be interpreted as `01,10,0,0,11`, which would give back the correct message `HELLO`, or as `0,11,0,0,0,11`, which would give the incorrect message `LOLLLO`.

Definition 19. We say that a code is **uniquely decodable** if each concatenation of codewords has a unique parsing into codewords, that is, if we can be sure of getting the correct message when decoding a sequence of codewords. The kinds of encodings that give uniquely decodable codes are the ones that are most interesting in information encoding.

Example 20. Let's have a look at a few different Γ mappings, and check whether they are uniquely decodable or not:

$\mathcal{A}$	Γ_O	Γ_A	Γ_B	Γ_C
a	00	0	0	0
b	01	01	10	01
c	10	10	110	011
d	11	11	1110	0111

FINISH THIS EXAMPLE

Definition 21. A code is said to be **prefix-free** if no codeword is the prefix of a longer codeword. Prefix-free codes are preferred in encoding information. They are also called **instantaneous codes**, since they allow instantaneous decoding.

Remark. Prefixes seem like a good idea, but the problem with codes that are not prefix-free is that when you concatenate words, you need to have all of them to be able to decode it correctly, as shown in the next example.

Example 22. Here's an example of why prefix codes are not the best in terms of decoding. The following code is uniquely decodable, but it uses a prefix:

$\mathcal{A}$	Γ
a	0
b	00001

If the decoder receives the sequence `00`, it cannot instantaneously determine wether this is the start of a `b` or two concatenated `a`, and so it has to wait until it has the full string of bits to be able to decode it correctly.

Theorem 23 (Kraft-McMillan 1)**.** *If a D-ary code is uniquely decodable, then its codeword lengths $l_1, l_2, \dots, l_M$ satisfy*

$$D^{-l_1} + \cdots + D^{-l_M} \leq 1 \quad \textit{(Kraft's inequality)} \tag{22}$$

Remark. Kraft's sum is only about lengths!

Example 24. Let's look at the following code:

$\mathcal{A}$	$\mathcal{C}$
`a`	`01`
`b`	`0101`

This is a 2-ary (binary) code with lengths 2 and 4, so Kraft's sum gives

$$2^{-2} + 2^{-4} = \frac{1}{4} + \frac{1}{16} = \frac{5}{16} \leq 1,$$

and Kraft's inequality is satisfied, however the code is clearly <u>not</u> uniquely decodable.

Summary.

- $\mathcal{C}$ is uniquely decodable $\implies$ it satisfies the kraft inequality
  contrapositive: $\mathcal{C}$ does not satisfy the Kraft inequality $\implies$ $\mathcal{C}$ is NOT uniquely decodable.
- the converse is NOT true.

28th February 2019

These following properties are what we aim for with any encoding map $\Gamma : \mathcal{A} \to \mathcal{C}$, in order to optimize transmission and decoding speed:

- $\mathcal{C}$ be uniquely decodable
- $\mathcal{C}$ be prefix-free
- $\mathcal{C}$ have its average codeword length be as small as possible

Theorem 25 (Kraft-McMillan 2)**.** *If $l_1, \dots, l_m$ satisfy Kraft's inequality for some positive integer D, then there exists a D-ary prefix-free code that has codeword lengths $l_1, \dots, l_m$.*

This second part of the Kraft-McMillan theorem implies that any uniquely decodable code can be substituted by a prefix-free code of the same codeword lengths.

Summary (Kraft-McMillan)**.** The theorem is in 2 parts:

1. If a D-ary code is uniquely decodable, then its codeword lenghts $l_1, \dots, l_M$ satisfy Kraft's inequality $D^{-l_1} + \cdots + D^{-l_M} \leq 1$.
2. If the positive integers $l_1, \dots, l_M$ satisfy Kraft's inequality for some integer D, then there exists a D-ary <u>prefix-free</u> code that has those codeword lengths.

Definition 26. We define the **average length** $L(S, \Gamma)$ as

$$L(S, \Gamma) = \sum_{S \in \mathcal{A}} p_S(s) \underbrace{l\big(\Gamma(s)\big)}_{\text{shorthand } l(s)}. \tag{23}$$

Sometimes we write

$$L(S, \Gamma) = \sum_i p_i l_i. \tag{24}$$

This is rather intuitively defined, as the expected length should according to common sense indeed depend on the length of each codeword, and on its probability of appearing in the code.
The units of the average length are **code symbols**. When $D = 2$, the units are **bits**.

Question. An interesting question we may now ask ourselves is the following: is there a lower bound to the average length for uniquely decodable codes?

Theorem 27 (lower bound on average length)**.** *Let Γ be the encoding map of a D-ary code for the source S. If the D-ary code is uniquely decodable, then*

$$H_D(S) \leq L(S, \Gamma). \tag{25}$$

The entropy is a lower bound to the average length.

Remark. A key observation we may make is that the definition of the average length is somewhat similar to that of the entropy:

$$L(S, \Gamma) = \sum_{s \in \mathcal{A}} p(s) l(\Gamma(s))$$

$$H_D(S) = \sum_{s \in \mathcal{A}} p(s) \log_D \left(\frac{1}{p_S(s)} \right)$$

In fact, the definitions are identical iff $l(\Gamma(s)) = \log_D \left(\frac{1}{p_S(s)} \right)$. Unfortunately this equality is often not possible (the log is often not an integer). But what if we chose $l(\Gamma(s)) = \left\lfloor \log_D \left(\frac{1}{p_S(s)} \right) \right\rfloor$? Is it a valid choice for a prefix-free code (is Kraft's inequality satisfied)?

Definition 28 (Shannon-Fano code)**.** A code $\mathcal{C}$ for which $l_i = \lceil -\log_D \big(P_S(s)\big) \rceil$ is called a **Shannon-Fano code**. Visually, it is constructed by going from the top down when creating the code tree.

For all s we $p_S(s)$

Remark. The Shannon-Fano code is not optimal (WHY? COMPLETE THIS).

Definition 29. We say that a probability distribution is **diadic** iff

$$p_i = D^{-l_i} \tag{26}$$

Theorem 30. *It is possible to make the codewords lengths l_i equal the entropy iff the code is diadic.*

CLEAR UP THE THING ABOUT THE -log(pi) THAT I DONT UNDERSTAND

Remark. Most probability distributions are not diadic. Then, $-\log(p_i)$ is not an integer, and we can't make the length equal that number.

Definition 31 (Huffman code)**.** Visually, the Huffman code on an alphabet is constructed by going from the bottom up when creating the code tree, and grouping together the least probable symbols.

Example 32. The Huffman code on $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$ is:

$\mathcal{A}$	Γ_H
a	000
b	001
c	01
d	1

We can compute the expected length and the entropy (in bits):

$$L(S, \Gamma_H) = 0.15 + 0.15 + 0.2 + 0.8 = 1.3$$
$$H_2(S) = \ldots = 1.022$$

5th March 2019

Theorem 33. *The average length can also be computed by adding together the probabilities of all nodes on the code tree, except for the last leaves:*

$$\underbrace{\sum_{\substack{i \in \\ terminal \\ leaves}}}_{L(S,\Gamma)} = \sum_{\substack{j \in \\ intermediate \\ nodes}} q_j. \tag{27}$$

Theorem 34 (optimality of Huffman codes)**.** *Let Γ_G be an encoder of a D-ary Huffman code for S, and let Γ be another D-ary uniquely decodable encoder for S. Then*

$$L(S, \Gamma_H) \leq L(S, \Gamma). \tag{28}$$

Basically, the Huffman code on S is the optimal code on S.

Definition 35 (IID source)**.** A source is said to be **IID** (Independant and Identically Distributed) iff COMPLETE THIS

Most sources are not IDD.

Example 36. Let $S_1, S_2 \in \{1, 2, \dots, 6\}$. They are independant and uniformly distributed. Let (L_1, L_2) be the first and second digit of the sum $S_1 + S_2$. We can compute $P_{L_1|L_2}(1|1) = \frac{P_{L_1|L_2}(1,1)}{P_{L_1}(1)}$. We know that the event $(L_1, L_2) = (1, 1)$ is the same as the event $S_1 + S_2 = 11$, which is the same as saying $(S_1, S_2) \in \{(5, 6), (6, 5)\}$, which has probability $2/36$. Then the conditional probability that we wanted to compute is

$$\frac{2/36}{3/36 + 2/36 + 1/36} = \frac{1}{3}$$

Definition 37 (conditional entropy)**.** Let p_X be the probability distribution of a random variable X. We already know how to compute the entropy $H(x)$ from this probability distribution. Then $p_{X|Y=y}$, which is also a probability distribution, allows us to compute the **conditional entropy** $H(X|Y = y)$:

$$H_b(X|Y = y) = -\sum_{x \in X} p_{X|Y}(x|y) \log_b\big(p_{X|Y}(x|y)\big) \tag{29}$$

Example 38 (continuation)**.** COMPLETE THIS FROM THE NOTES :DDDDDDDDDDDDDDD I AM GOING INSANE AND 2 HOURS LEFT :DDDDDDDDDDD

7th March 2019

IS THE ENTROPY EQUAL TO THE AVERAGE LENGTH IFF IT IS A HUFFMAN CODE?

Theorem 39. *Let X and Y be two random variables. Then*

$$H(X|Y) \leq H(x), \tag{30}$$

with the equality iff X and Y are independent.

Theorem 40 (chain rule of entropy)**.** *Let $S_1, \dots, S_n$ be random variables. Then*

$$\boxed{H(S_1, \dots, S_n) = \sum_{i=1}^{n} H(S_1|S_1, \dots, S_{i-1})} \tag{31}$$

For this to make a bit more sense, recall that $P_{X,Y}(x, y) = P_X(x)P_{Y|X}(y|x)$ (this follows directly from the definition of conditional probability). More generally, $P_{S_1,\dots,S_n}(s_1, \dots, s_n) = \prod_{i=1}^{n} P_{S_i|S_1,\dots,S_{i-1}}(s_i|s_1, \dots, s_{i-1})$. The chain rule of entropy is very similar.

This equality will help us in proving many theorems.

Example 41 (continuation)**.** $H(l_1) = 0.65$ bits, $H(l_2) = 3.2188$ bits, $H(l_1, l_2) = 3.744$ bits, $H(l_2|l_1) = H(l_1, l_2) - H(l_1) = 2.624$ bits (as obtained before).

Example 42 (continuation)**.** We saw that (S_1, S_2) determine (l_1, l_2). Then what is $H(l_1, l_2|S_1, S_2)$? We can know that $H(l_1, l_2|(S_1, S_2) = (s_1, s_2)) = 0$. This is because (S_1, S_2) fully determine (l_1, l_2) (they are NOT independent). As such, when we know that $(S_1, S_2) = (s_1, s_2)$, we know exactly and can *predict* the only possible digits (l_1, l_2) of

the sum of s_1 and s_2, which means that there is no entropy here.
Suppose we know $H(l_1, l_2)$ and $H(S_1, S_2)$. Then we can compute $H(S_1, S_2|l_1, l_2)$:

$$H(S_1, S_2, l_1, l_2) \overbrace{=}^{\substack{\text{chain}\\\text{rule}}} \begin{cases} H(S_1, S_2) + \cancel{H(l_1, l_2|S_1, S_2)} \\ H(l_1, l_2) + H(S_1, S_2|l_1, l_2) \end{cases}$$

$$\implies H(S_1, S_2, |l_1, l_2) = H(S_1, S_2) - H(l_1, l_2), \quad \text{which we both know by supposition.}$$

Example 43. Let's say we have a random variable X which takes a value in $\{0, +1, -1, +2, \dots, +13, -13\}$. Suppose X is uniformly distributed. Any weighing strategy (not yet determined) is an encoding $\Gamma : \mathcal{A} \to \mathcal{C}$ ($|\mathcal{C}| = 3$, so this is a 3-ary code). Γ is a bijective function $x \leftrightarrow s_1 s_2 \dots s_L$. So $H_3(x) = H_3(S_1, S_2, \dots S_L)$.

NOTES: a negative number means "light" on the balance, a positive number means it is "heavy". At each step of the guessing process, he makes sure that the new balance is independent of previous knowledge, and that it is evenly distributed (even chances of going on each side). Actually the balls are billiard. One coould be heavier or lighter, ut not the 0 ball. You have a balance for balls. How many times to weigh to determine if one bakk us fakse and if yes, which one and settle if it is heavier or lighter.

12th March 2019

So far, we have assumed that we have a random variable over the alphabet $\mathcal{A}$, an encoding function Γ which is a map $\mathcal{A} \to \mathcal{C}$, from the alphabet to the code (which was assumed to be uniquely decodable), and a source outputting a sequence of n random variables. We saw that if this source is IID, then the total length divided by n will tend to $L(S, \Gamma)$ as $n \to \infty$. We saw that the entropy is a lower bound to the average length.

We will now drop the assumption that the source is IID: this is very common, for example, when you try to compress voice, or video, or any kind of "natural" information.

Example 44. We reuse the example of the coin flip source: $S_i \in \{H, T\}$. $S_1, S_2, \dots, S_n$ are IID. Therefore

$$p_{S_1, S_2, \dots, S_n}(s_1, s_2, \dots, s_n) = \prod_{i=1}^{n} p_{S_i}(s_i) = \left(\frac{1}{2}\right)^n$$

Definition 45. The source $\mathcal{S} = S_1 S_2 \dots S_n$ is said to be **regular** iff

1. $H(\mathcal{S}) = \lim_{i \to \infty} H(S_i)$ (the entropy per symbol) exists, and
2. $H^*(\mathcal{S}) = \lim_{i \to \infty} H(S_i | S_1, \dots, S_{i-1})$ (the entropy rate) exists.

Example 46 (cont.)**.** The coin flip source is regular:

$$H(\mathcal{S}) = 1 = H^*(\mathcal{S})$$

Example 47 (Sunny-Rainy source)**.** $S_i \in \{S, R\}$ represents the weather on day i. S_i is uniformly distributed. If the weather one way one day, it will stay the same tomorrow with probability q, and change with probability $1 - q$.

$$p_{S_1, \dots, S_n}(s_1, \dots, s_n) = p_{S_1}(s_1) \cdot \prod_{i=2}^{n} p_{S_i|S_{i-1}}(s_i|s_{i-1}).$$

For example, $p_{S_1, S_2, S_3, S_4}(RRSR) = \frac{1}{2} \cdot q \cdot (1-q) \cdot (1-q)$. More generally, $p_{S_1, \dots, S_n}(s_1, \dots, s_n) = \frac{1}{2} \cdot (1-q)^c \cdot q^{n-1-c}$, where c is the number of transitions.

We may also check wether the source is regular or not. $H(S_i) = H(S_1) = 1$, therefore the entropy per symbol exists and equals 1. Let us also compute $H(S_i|S_{i-1})$. We know that $H(S_i|S_{i-1} = R) = h(q)$ and $H(S_i|S_{i-1} = S) = h(1-q) = h(q)$ (where h is the binary entropy function, that is, the entropy function adapted to binary bernouilli trials, which is intuitively symmetrical with regards to 1/2). Finally $H(S_i|S_{i-1}) = h(q) = H(S_1|S_{i-1}, \dots, S_1)$ (the outcomes before the previous day do not have any effect). Therefore the entropy rate exists, and is equal to $h(q)$.

Definition 48. A Markov chain is the simplest kind of source which has some memory: each random variable depends only on the state of the one immediately preceding it.

Remark. The source in example 0.7 is a Markov chain.

Definition 49. A source $S_1, S_2 \ldots, S_i \in \mathcal{A}$, is **stationary** if for every n, k positive integers, the statistic of $(S_1, \ldots, S_n)$ is the same as the statistic of $(S_{k+1}, \ldots, S_{k+n})$. Formally,

$$p_{S_1,\ldots,S_n}(s_1, \ldots, s_n) = P_{S_{k+1},\ldots,S_{k+n}}(s_1, \ldots, s_n). \tag{32}$$

Theorem 50. *A stationary source is always regular.*

Theorem 51. *For a stationary source $\mathcal{S}$,*

$$H^*(\mathcal{S}) \leq H(\mathcal{S}), \tag{33}$$

with equality iff the symbols are independent.

14th March 2019

Question. What if, instead of encoding codewords one at a time, we encoded concatenations of codewords?

$$\boxed{H_D(S) \leq L(S, \Gamma_H) \leq L(S, \Gamma_{SF}) < H_D(S) + 1} \tag{34}$$

This inequation is not directly related but muchho importanto in source coding. We can tho adapt it to block-encoding (idk how it's called):

$$H_D(S_1 \ldots S_n) \leq L(\mathcal{S}, \Gamma_H) \leq L(\mathcal{S}, \Gamma_{SF}) < H_D(S_1 \ldots S_n) + 1 \tag{35}$$

If we divide everything by n to get the average codeword length per symbol:

$$\frac{H_D(S1 \ldots S_n)}{n} \leq \frac{L(\mathcal{S}, \Gamma_H)}{n} < \frac{H_D(S_1 \ldots S_n)}{n} + \frac{1}{n} \tag{36}$$

The $\frac{1}{n}$ goes to 0 as n grows large.

Our goals: study the behavior of $\frac{H_D(S_1 \ldots S_n)}{n}$ as n grows large and try to relate it to $H_D^*(\mathcal{S})$ (entropy rate).

Example 52. Consider a monkey source S that randomly picks one letter at a time from a French book. $H(S) =$ 3.95 bit2 s, so a Huffman code Γ_H approaches $L(S, \Gamma_H) \approx 4$ bits/letter when encoding French monkey text.

However, a Lempel-Ziv code (used by various compression programs) approaches 1 bit per letter when compressing French text. This is due to the fact that letters in a French text are not completely random, therefore the entropy goes down by quite a bit, and beat the Huffman code.

This means that as n grows large, $\frac{H_D(S_1 \ldots S_n)}{n} \to 1$. This, rather than $H(S)$, is the important quantity for us.

If $S_1 \ldots S_n$ were IID, then

$$\frac{H_D(S_1 \ldots S_n)}{n} = \frac{H_D(S_1) \cdot H_D(S_2) \ldots H_D(S_n)}{n} = H(S_1) \quad \forall 1 \leq i \leq n$$

Imagine you start with the text from a book, and take the alphabet of that book (all symbols that appear in it). You put these in a table and assign an integer to each (the position in the table), in binary. You then start looking for groups of 2 letters, and if you find some, you add them to the dictionary, so n increases (it has a limit). Same thing for 3 symbols, or groups that appear often.

Theorem 53 (Cesàro mean)**.** *Consider a source of real-valued numbers $a_1 \ldots a_n$. If $\lim_{n\to\infty} a_n = A$, then if $c_n = \frac{a_1 + a_n}{n}$, we have that $\lim_{n\to\infty} c_n = A$.*

Theorem 54. *Let S be a source. Then*

1. *if S is stationary, then S is regular,*
2. *$\frac{H_D(S_1 \ldots S_n)}{n}$ is non-increasing in n,*
3. *$\lim_{n\to\infty} \frac{H_D(S_1 \ldots S_n)}{n} = H_D^*(S)$.*

Summary. Let S be a stationary source outputting an infinite sequence of symbols $S_1 \ldots S_n$. By encoding blocks of n keywords at a time using a D-ary code, the average codeword length per symbol approaches $H^*(S)$ as n grows large. No uniquely decodable D-ary code can do better than $H^*(S)$.

Example 55 (The 20 Questions Game)**.** Was a very popular game on UK and US TV. You ask questions where the answers are yes or no. Equivalent to a binary search (akinator).

Let X be a random variable. Question: how many YES/NO questions do we need to find the realization of X, how should we ask the questions?

The idea is to build a binary code Γ for X. Once Γ is fixed, identify the realization of $X (= \bar{X})$, which is equivalent to finding $\Gamma(\bar{X})$. The i-th question should reveal the i-th bit of $\Gamma(\bar{X})$. The average number of questions should be the average codeword length of Γ. If Γ is Γ_H, we cannot do better in terms of the average number of questions.

Let us consider a random variable X with $\mathcal{A} = \{a, b, c, d, e\}$, $p(a) = 0.1, p(b) = 0.1, p(c) = 0.2, p(d) = 0.2, p(e) = 0.4$. After its construction, the Huffman code is $a = 000, b = 001, c = 010, d = 011, e = 1$. Let's ask some questions from the top of the tree:

1. is $X = e$? NO ==¿ the first bit of X is 0.
2. is $X \in \{c, d\}$? NO ==¿ the second bit of X is 0.
3. is $X = b$? YES ==¿ the third bit of X is 1.

Therefore, $X = 001 = b$.

Let us now discuss the optimality of this strategy. We have seen that a binary code (prefix-free) implies a question strategy, and that Γ_H gave the best strategy, since the average number of questions is equal to the average codeword length of Γ.

Question: does a questioning strategy (YES/NO) as above always lead to a prefix-free code?

We start with the root: let $X \in \mathcal{X}$ be a random variable. Split $\mathcal{X}$ into $\mathcal{A}$ and $\mathcal{A}^c$ (complement) and suppose that $X \in \mathcal{A}$. When we ask the question, we will know in which of the two it is. Suppose that the answer is yes: then $X \in \mathcal{A}$. Continue by splitting $\mathcal{A}$ into $\mathcal{B}$ and $\mathcal{B}^c$ and repeat the process. By construction, we obtain a prefix-free code.

ENCODING OF INTEGERS: we saw that the Lempel-Ziv code needed to encode integers. Let's think about some binary prefix-free codes for positive integers.

The first one is the "natural" way, that is, encoding each number into its binary representation. However, this is not good, because it is not prefix-free, and has $l(n) = \lfloor \log(n) \rfloor + 1$.

The next try (elias code 1) would be adding $l(n) - 1$ zeros in front of each codeword n. This is prefix-free, but it is pretty long: $l(n) = 2\lfloor \log(n) \rfloor + 1$.

The next try (elias code 2) would be replacing the leading zeros and the following one by $c_1(l(n)$ (c_1 = elias code 1). Then we get a ITS PI DAY PEOPLE

3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808132823066470

Summary (Source Coding)**.**

- If a D-ary code is UD for source S,

  $$H_D(S) \leq L(S, \Gamma) < H_D(S) + 1$$

  The first inequality comes from Kraft's sum and the IT inequality.
  This also applies to $\bar{S} = S_1 S_2 \dots S_n$:

  $$\begin{aligned} H_D(\bar{S}) \leq L(\bar{S}, \Gamma) < H_D(\bar{S}) + 1 \iff \\ \frac{H_D(\bar{S})}{n} \leq \frac{L(\bar{S}, \Gamma)}{n} < \frac{H_D(\bar{S})}{n} + \frac{1}{n} \end{aligned}$$

  (we divided by n to get the entropy per symbol). The second and last terms (without $\frac{1}{n}$, which tends to 0 as $n \to \infty$) tend to $H^*(\bar{S})$ if the source is stationary.

- For encoding integers, we saw the Elias code (UD), which encodes $n \in \mathbb{N}$ with $\approx \log_2 n + 1$ bits.
  Among other things, we have left out what's called "universal source coding" (example in the notes).