5. Communication theory

Contents

(All variables are real and one-dimensional unless otherwise specified.)

1. Entropy as information

Consider a **discrete** random variable X that picks a value from $\{x_1, x_2, \ldots, x_n\}$ with probabilities $\{p_1, p_2, \ldots, p_n\}$. Shannon defined the variable's "**information**" as

$$H(X) = -\sum_i p_i \log(p_i) \,,$$

where the logarithm can use any base as a change in base only rescales the quantity. The default base is 2, making the unit of $H(X)$ a **bit** (i.e. "**b**inary dig**it**"). Another common base is e (for easier calculus), and the corresponding unit is a **nat** (i.e. "**na**tural digi**t**"). Sometimes a base 10 may be used as well, with its unit called a **dit** (i.e. "**d**ecimal dig**it**").

1.1 Interpretation

How can we interpret the definition, though? After rearranging, we can get

$$H(X) = \left\langle \log\left(\frac{1}{p}\right) \right\rangle \,,$$

which is the expected value of some quantity. In fact, we may call $1/p_i$ as the "**surprisal**" of x_i, i.e. the extent of how surprised you get when you observe $X = x_i$. If p_i is small, you are very surprised to observe $X = x_i$, and

a more surprising event is more informative.

(Imagine I tell you that something you believe is actually false.) Hence, $H(X)$ tells you how surprised you expect to get after measuring X and thus how informative the measurement is.

Shannon used logarithm in the definition because it desirably turns multiplication of independent events' probabilities into addition of their respective information. While he invented it purely on mathematical grounds, Shannon's quantity turns out to identical to **entropy** in statistical physics, which intends to measure how "random", "disordered", or "chaotic" a physical system is. This coincidence reminds physicists of **Maxwell's demon (https://en.wikipedia.org/wiki/Maxwell%27s_demon)**, an age-long mysterious thought experiment regarding entropy, and brings out a profound philosophical question:

is information physical?

Trivia. Shannon decided to rename $H(X)$ as "entropy" after discussing with von Neumann:

> *My* [Shannon's] *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 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.*

1.2 Example: tossing a coin

What is the entropy of tossing a coin, be it fair or not?

Solution. Let X be a coin's outcome (head or tail) and p the probability for it to be a head. The entropy is

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

This is the formula of entropy of all binary variables. As this formula appears a lot in communication theory, we conventionally define the **binary entropy function** as

$$\mathrm{H_b}(p) \equiv -p\log p - (1-p)\log(1-p)\,.$$

Defining $0\log 0 \equiv 0$, the function ranges from 0 to 1. On one hand, it attains its maximum at $p = 0.5$, meaning that tossing a fair coin gives us a lot of information (i.e. collapsing from two equiprobable options to one exact outcome). On the other hand, it attains its minimum at $p = 0$ or $p = 1$, meaning that tossing a completely biased coin tells us nothing new as we are already sure about its outcome before tossing.

1.3 From probability to entropy

Consider two random variables X and Y. Their **joint entropy** assesses the total information gained from measuring them together.

$$H(X,Y) = -\sum_x\sum_y P(X=x,Y=y)\log P(X=x,Y=y)$$

As the variables may be correlated, the **conditional entropy** of Y on X assesses the extra information that Y provides after knowing X.

$$\begin{aligned} H(Y \mid X) &= -\sum_x \sum_y P(X=x, Y=y) \log P(Y=y \mid X=x) \\ &\equiv H(X,Y) - H(X) \end{aligned}$$

From these two notions we can define **mutual information** between X and Y.

$$\begin{aligned} I(X;Y) &= -\sum_x \sum_y P(X=x, Y=y) \log \frac{P(X=x)P(Y=y)}{P(X=x, Y=y)} \\ &\equiv H\left(X\right) + H\left(Y\right) - H\left(X,Y\right) \\ &\equiv H\left(Y\right) - H\left(Y \mid X\right) \equiv H\left(X\right) - H\left(X \mid Y\right) \end{aligned}$$

Note that a semicolon is used in $I(X;Y)$ instead of a comma so that we can have, for example, $I(X,Y;Z) = H(X,Y) - H(X,Y \mid Z)$. The Wikipedia article "**Quantities of information (https://en.wikipedia.org/wiki/Quantities_of_information)** " summarizes the relationships between various kinds of entropy.

After all, Shannon developed "entropy" for his theory of communication. In his pioneering paper, ***A Mathematical Theory of Communication*** **(https://people.math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf)** , Shannon depicted "communication" with the following schematic diagram.

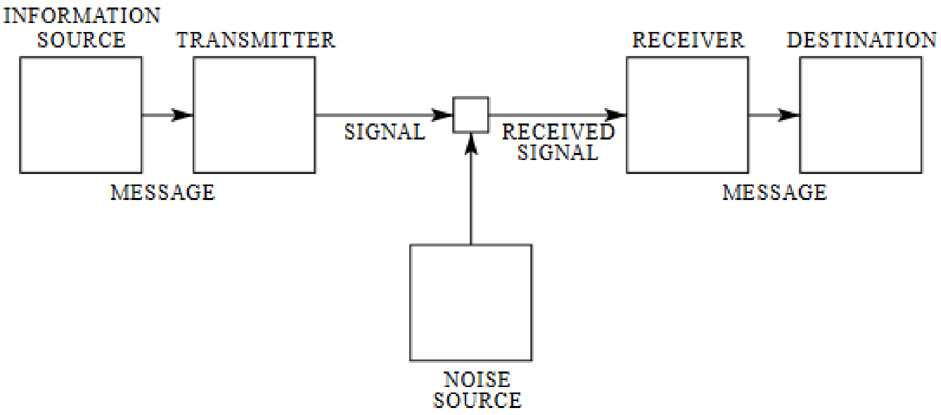

Fig. 1 Shannon's model of communication**.**

2. Data transmission

As Alice sends a message to Bob, they form a **communication channel** in which Alice is the **transmitter** and Bob is the **receiver**. Letting X and Y respectively be the messages that Alice sends and Bob receives, their **channel capacity** is defined as the maximum of their mutual information $I(X;Y)$ over all possible distributions of X (in simple words, over all possible messages that Alice sends):

$$C = \max_{P(X)} I(X;Y)\,.$$

It measures the maximum amount of information that Alice can send to Bob reliably.

2.1 Example: binary symmetric channel

The simplest type of channel is a **binary symmetric channel** (BSC). Alice sends out $X = 0$ with probability a and $X = 1$ otherwise, then Bob correctly receives $Y = X$ with probability p but accidentally $Y = 1 - X$ otherwise. The channel is symmetric because the probability of a correct reception does not depend on the input.

How much mutual information $I(X;Y)$ do Alice and Bob share in the BSC? Hence what is its capacity C?

Solution. Define $q = 1 - p$ and $b = 1 - a$. We have $\begin{cases} P(X=0) = a \\ P(X=1) = b \end{cases}$ and $\begin{cases} P(Y=0 \mid X=0) = P(Y=1 \mid X=1) = p \\ P(Y=1 \mid X=0) = P(Y=0 \mid X=1) = q \end{cases}$, so $P(Y=0) = ap + bq$, and the mutual information is

$$\begin{aligned} I(X;Y) &= H(Y) - H(Y \mid X) \\ &= \mathrm{H_b}(ap+bq) - aH(Y \mid X=0) - bH(Y \mid X=1) \\ &= \mathrm{H_b}(ap+bq) - a\mathrm{H_b}(p) - b\mathrm{H_b}(p) \\ &= \mathrm{H_b}(ap+bq) - \mathrm{H_b}(p)\,. \end{aligned}$$

To get the capacity, we need to solve $\dfrac{\partial}{\partial a} I(X;Y) = 0$ for the optimal a^* and then substitute it back into $I(X;Y)$. After some algebra, we can get

$$C = 1 - \mathrm{H_b}(p)\,.$$

2.2 Example: two binary symmetric channels

Upon interpreting Alice's message X, Bob from last section sends his message Y to another receiver Charlie. Charlie gets $Z = Y$ with probability $p' \equiv 1 - q'$ but $Z = 1 - Y$ otherwise. How much mutual information $I(X;Z)$ do Alice and Charlie share? Hence what is their channel's capacity C'?

Solution. Notice that

$$\begin{aligned} P(Z=0 \mid X=0) &= \sum_{y\in\{0,1\}} P(Z=0 \mid Y=y)P(Y=y \mid X=0) \\ &= pp' + qq' \end{aligned}$$

and other conditional probabilities can be similarly calculated. In fact, Alice sends her message to Charlie successfully with probability $u \equiv pp' + qq'$ but fails otherwise, so we can apply last section's results to obtain $I(X;Z) = \mathrm{H_b}[au + b(1-u)] - \mathrm{H_b}(u)$and $C' = 1 - \mathrm{H_b}(u)$.

2.3 Example: multiple binary symmetric channels

Charlie from last section continues to pass the message to other people, thus forming a chain of communication. Each transmission succeeds with an agent-independent chance $p \equiv p'$.

Let $X^{(n)}$ be the message received by the nth person in the chain, whereas Alice's message is denoted by $X^{(0)}$. If Alice sends out $X^{(0)} = 0$ with probability a, how much mutual information $I\left[X^{(0)};X^{(n)}\right]$ does she share with the nth person? And what is their channel capacity $C^{(n)}$?

Solution. Let $\mathbf{P}^{(k)} \equiv \begin{pmatrix} P\left[X^{(k)}=0\right] \\ P\left[X^{(k)}=1\right] \end{pmatrix}$ represent the distribution of the kth person's message $X^{(k)}$. While $\mathbf{P}^{(0)} = \begin{pmatrix} a \\ b \end{pmatrix}$, Section 2 tells us that $\mathbf{P}^{(1)} = \begin{pmatrix} ap+bq \\ aq+bp \end{pmatrix}$. (Again, $q \equiv 1-p$ and $b \equiv 1-a$.) In fact, we can regard each BSC as a **channel matrix** $\mathbf{B} = \begin{pmatrix} p & q \\ q & p \end{pmatrix}$ so that $\mathbf{P}^{(1)} = \mathbf{B}\mathbf{P}^{(0)}$, and an iteration over this equation leads to $\mathbf{P}^{(n)} = \mathbf{B}^n\mathbf{P}^{(0)}$. Through comparison with $I\left[X^{(0)};X^{(1)}\right] = \mathrm{H_b}(ap+bq) - \mathrm{H_b}(p)$, we conclude that

$$I\left[X^{(0)};X^{(n)}\right] = \mathrm{H_b}\left(\left[\mathbf{P}^{(n)}\right]_0\right) - \mathrm{H_b}\left[(\mathbf{B}^n)_{00}\right],$$

where $\left[\mathbf{P}^{(n)}\right]_0$ and $(\mathbf{B}^n)_{00}$ are the top and the top-left entries of $\mathbf{P}^{(n)}$ and $\mathbf{B}^n$. The channel capacity is similarly argued to be $C^{(n)} = 1 - \mathrm{H_b}\left[(\mathbf{B}^n)_{00}\right]$.

Mathematics. You can of course compute $\mathbf{B}^n$ with brute force, but linear algebra suggests a more elegant method, namely **diagonalization**. Recall your knowledge in linear algebra: because $\mathbf{B}$ is real symmetric,

$$\mathbf{B} = \mathbf{V}^{-1}\mathbf{D}\mathbf{V} \equiv \mathbf{V}^{-1}\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}\begin{pmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{pmatrix}$$

for its eigenvalues λ and corresponding **orthonormal** eigenvectors $\mathbf{v}$. The orthonomality means $\mathbf{v}^\top\mathbf{v} = 1$ and implies $\mathbf{V}^{-1} = \mathbf{V}^\top$, so

$$\begin{aligned}\mathbf{B}^n &= \left(\mathbf{V}^{-1}\mathbf{D}\mathbf{V}\right)^n = \mathbf{V}^{-1}\mathbf{D}^n\mathbf{V} = \mathbf{V}^\top\mathbf{D}^n\mathbf{V}\\ &= \begin{pmatrix}\mathbf{v}_1^\top\\ \mathbf{v}_2^\top\end{pmatrix}\begin{pmatrix}\lambda_1^n & 0\\ 0 & \lambda_2^n\end{pmatrix}\begin{pmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{pmatrix}.\end{aligned}$$

Since the eigenvalues of $\mathbf{B}$ are $\lambda_1 = 1$ and $\lambda_2 = p - q$, which correspond to $\mathbf{v}_1 = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}$ and $\mathbf{v}_2 = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}$, we get $\mathbf{B}^n = \frac{1}{2}\begin{bmatrix}(p-q)^n+1 & 1-(p-q)^n\\ 1-(p-q)^n & (p-q)^n+1\end{bmatrix}$. Hence, if people keep sending Alice's message, the probability for Alice's message to be correctly transmitted saturates at $\lim\limits_{n\to\infty}\left(\mathbf{B}^n\right)_{00} = \frac{1}{2}$, i.e. as uncertain as tossing a fair coin.

2.4 Example: binary asymmetric channel

Consider Alice and Bob from Section 2.1 again. Their channel is now modified to have $\begin{cases}P(Y=0\mid X=0)=p_0\\ P(Y=1\mid X=0)=1-p_0\equiv q_0\end{cases}$ and $\begin{cases}P(Y=0\mid X=1)=1-p_1\equiv q_1\\ P(Y=1\mid X=1)=p_1\end{cases}$; in other words, the channel matrix becomes $\mathbf{B} = \begin{pmatrix}p_0 & q_1\\ q_0 & p_1\end{pmatrix}$.

This is a **binary asymmetric channel** (BAC) because the probability of successful transmission depends on the input. Consequently, what is their new mutual information $I(X;Y)$?

Solution. Because $P(Y=0) = ap_0 + bq_1$,

$$\begin{aligned}I(X;Y) &= H(Y) - H(Y\mid X)\\ &= \mathrm{H_b}(ap_0+bq_1) - aH(Y\mid X=0) - bH(Y\mid X=1)\\ &= \mathrm{H_b}(ap_0+bq_1) - a\mathrm{H_b}(p_0) - b\mathrm{H_b}(p_1).\end{aligned}$$

Challenge. It requires cumbersome differentiation to calculate the corresponding channel capacity, which amounts to

$$C = \log\left[1+\left(\frac{1}{2}\right)^{-\frac{\mathrm{H_b}(q_1)-\mathrm{H_b}(q_0)}{p_0-q_1}}\right] - \frac{p_0\mathrm{H_b}(q_1)-q_1\mathrm{H_b}(q_0)}{p_0-q_1}.$$

2.5 The channel-coding theorem (https://en.wikipedia.org/wiki/Noisy-channel_coding_theorem)

But why should we be concerned by a channel's capacity?

Let us consider the process depicted in Fig. 1 again. Before she sends a message M, the transmitter Alice first **encodes** it with a function e so that it becomes a **code** $X = e(M)$ that can pass their channel. The receiver Bob then receives a possibly distorted signal Y and **decodes** it with a function d to obtain a copy of the message $M' = d(Y)$, which may be distorted as well. In this case, can Bob trust M'?

The **channel-coding theorem** says that the answer depends on their channel capacity C and some other factors.

- On one hand, the codes X and Y are expressed in **code symbols**, which are restricted to be $\{0, 1\}$ in our discussion.
- On the other hand, Alice writes her message M in **source symbols** from a **source language** Σ. As each source symbol σ occurs with probability p_σ, the **source entropy** is $H(\Sigma) = -\sum_{\sigma\in\Sigma} p_\sigma \log{(p_\sigma)}$.
- At the same time, Alice's hands are not infinitely fast, and she can only write R symbols per unit time.
- For similar physical reasons, the channel cannot transmit infinitely many but only S symbols per unit time.

The theorem states that if Alice's **entropy rate** $RH(\Sigma)$ is less than the channel's entropy rate SC, there exist functions (e, d) such that $M' = M$, so

Bob can know Alice's message perfectly despite noisy communication
as long as Alice generates information slower than the channel can transmit.

Conversely speaking, if $RH(\Sigma) > SC$, Bob can never fully retrieve Alice's message. (The theorem has several other name; in fact, Shannon called it "the fundamental theorem for a discrete channel with noise" in his paper.)

3. Data compression

In Fig. 1, the noise source only disrupts the communication between the transmitter and the receiver, whereas the communication between the information source and the transmitter is noiseless, so is that between the receiver and the destination. Data can pass to and fro without distortion in such **noiseless channels**, and communication therein can be made more efficient by techniques of **data compression**.

3.1 The source-coding theorem (https://en.wikipedia.org/wiki/Shannon%27s_source_coding_theorem)

The **source-coding theorem** is another important theorem from Shannon's paper, while he called it "the fundamental theorem for a noiseless channel". It concerns how efficiently data can be compressed. It can be stated in several equivalent ways, and here is a more intuitive version.

Suppoe we have encoded a message M with a function e to obtain a code $X = e(M)$, which is again assumed to be binary for simplicity. If no information is lost, the coding process is **lossless**, and we can perfectly retrieve $M = d(X)$ with some decoding function d. The theorem states the relationship between the code's **expected length** $|X|$ and the message's entropy $H(M)$ in a lossless proess.

$$H(M) \quad \overset{\forall e}{\leq} \quad |X| \quad \overset{\exists e}{\leq} \quad H(M) + 1$$

In particular, if M is written in n symbols from a source language Σ whose entropy is $H(\Sigma)$, we have $H(M) = nH(\Sigma)$, and the sandwich inequality consequently suggests $|X| = nH(\Sigma)$ as $n \to \infty$. Since $H(\Sigma) \leq 1$, the message's length decreases from n after coding, implying compression.

A coding process is, in contrast, **lossy** if some information is lost, and thus the resultant code X is of course smaller than the original message M. But keep in mind that "lossy" does not mean "useless"; for example, JPG and MP3 are very useful algorithms of lossy compression.

3.2 Example: coding algorithms

So which encoding functions e can yield $|X| \leq H(M) + 1$? Shannon code, Fano code, and Huffman code are three typical examples. In particular, Huffman code can be proved to be the best function among all possible e because it compresses data the most. (They are not mathematicians' functions becasuse they may map one message to several equally valid codes. They are just programmers' functions.)

Let's compare them with a toy example. If a source language Σ has two source symbols ↑ and ↓ that occur with probabilities $\begin{cases} p_\uparrow = 0.8 \\ p_\downarrow = 0.2 \end{cases}$, how do the three functions encode a message that consists of $n = 3$ symbols?

Solution. There are $2^n = 8$ possible messages. The table below sorts them in the descending order of their probabilities p and summarizes their codes, which are generated according to the following algorithms.

- Shannon first lets $\lceil -\log_2 p \rceil$ be the length of his codes, then he assigns a smaller code (i.e. a smaller binary number) to a more probable message while observing the requirement of a **prefix code**, meaning that no code can be the initial part of another code.
- Fano first divides the sorted table into a left half and a right half so that they are as equally probable as possible, then he appends 0 to the ends of the codes that correspond to the messages on the left but 1 to those on the right. He repeats this process until a code has been assigned to every message.
- Huffman first appends 0 to the head of the second least probable message but 1 to that of the least probable one, then he merges the two messages as one new message so that their

probabilities sum up. He also repeats this process until a code has been assigned to every message.

	↑↑↑	↑↑↓	↑↓↑	↓↑↑	↑↓↓	↓↑↓	↓↓↑	↓↓↓
p	0.512	0.128	0.128	0.128	0.032	0.032	0.032	0.008
$\lceil -\log_2 p \rceil$	1	3	3	3	5	5	5	7
Shannon	0	100	101	110	11100	11101	11110	1111100
Fano	0	100	101	110	11100	11101	11110	11111
Huffman	0	100	101	110	11100	11101	11110	11111

In this case, the Fano code happens to be as efficient as the Huffman code, whereas the Shannon code is just slightly worse. But even the Shannon code yields an expected length

$$|X| = 1 \times 0.512 + 3 \times 0.128 \times 3 + 5 \times 0.032 \times 3 + 7 \times 0.008 = 2.200$$

closer to its lower limit $H(M) = nH(\Sigma) \approx 2.166$ than to $n = 3$. Note that the codes presented here are just one possibility. You can obtain equally valid codes by, for example, flipping all 0's to 1's and all 1's to 0's.

Trivia. Huffman took a doctoral course taught by Fano, who had been Shannon's collaborator, at MIT, and Huffman came up with his algorithm while doing a question assigned by Fano.