QUANTUM INFORMATION

IAN LIM
LAST UPDATED JANUARY 25, 2019

These notes were taken for the *Quantum Information* course taught by Nilajana Datta at the University of Cambridge as part of the Mathematical Tripos Part III in Lent Term 2019. I live-TEXed them using Overleaf, and as such there may be typos; please send questions, comments, complaints, and corrections to itel2@cam.ac.uk.
Many thanks to Arun Debray for the LATEX template for these lecture notes: as of the time of writing, you can find him at https://web.ma.utexas.edu/users/a.debray/.

Contents

Lecture 1.

Friday, January 18, 2019

Note. Here's the relevant admin content for the first day. The lecturer's email is n.datta@damtp.cam.ac.uk, and course notes can be found on the CQIF website under Part III lectures.

Quantum information theory (QIT) was born out of classical information theory (CIT).

Definition 1.1. Classical information theory is the mathematical theory of information processing tasks, e.g. storage, transmission, processing of information.

In contrast, quantum information theory asks how these tasks can be performed if we harness quantum mechanical systems as information carriers. Such systems include electrons, photons, ions, etc.

QM has some novel features which are not present in our old Newtonian theories. We know that quantum systems obey the Heisenberg uncertainty principle, that energy is quantized in these systems, and QM systems cannot generically be copied (the famous no-cloning theorem). Quantum mechanically, one can describe the full state of a system without knowing the state of the subsystems– this is essentially the idea of entanglement.[1]

Here's a quick overview now of the structure of the course.

- Basic concepts of CIT
- Study of open quantum systems
- Mathematical tools for QIT
- Entanglement
- QIT itself

When we say open quantum systems, we mean quantum systems which interact with a broader environment. If we prepare a state and allow it to interact, what happens to the information stored in that state?

Classical information theory Historically, CIT was invented in 1948 with a pioneering paper by Claude Shannon. In this paper, he asked two critical questions.

Q1. What is the limit to which information can be *reliably* compressed?
Q2. What is the maximum rate at which information can be reliably sent through a communication channel?

That is, we may ask about how to encode information in such a way that it can still be recovered with a high probability of success. And we can ask how to send this information when our communication channels will naturally be noisy. The answers to these questions are known as *Shannon's Source Coding Theorem* and *Shannon's Noisy Channel Coding Theorem*, respectively.

[1] If you like, some composite states in a tensor product space cannot be decomposed into a direct product.

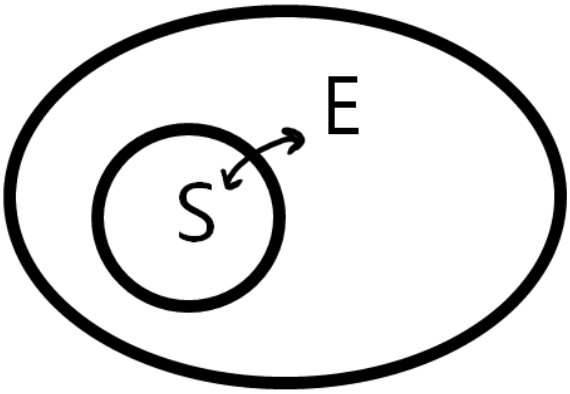


FIGURE 1. A sketch of the sort of systems we will be interested in in this class. We have an open system S which will naturally interact with its environment E.

What is information? We have an intuitive sense of what information means, but to formalize this takes a little work. In the loosest sense, information is associated to uncertainty and in particular information gain is related to a reduction in uncertainty.

Example 1.2. Suppose I have a system which takes some discrete values, e.g. I roll a fair die. The outcome is a variable x which takes values in some set, $J = \{1, 2, \ldots, 6\}$. We write that capital X is proportional to $p(x), x \in J$, where $P(X = x) = p(x) = 1/6 \, \forall x \in J$. That is, there is a probability mass function associated to the possible outcomes. The probability that we measure the system X in outcome x is $1/6$ for any outcome x in the set of outcomes.

We also define the following quantity.

Definition 1.3. *Surprisal* is the quantity

$$\gamma(x) = -\log p(x). \tag{1.4}$$

When an event is very unlikely and it happens anyway... you are very surprised. For example, $p(x) = 1 \implies \gamma(x) = 0$ (certainties are not very surprising) while $p(x) \approx 0 \implies \gamma(x)$ large. See Fig. **??** for a plot of γ versus p.

This quantity has some features:

- It only depends on $p(x)$ and not on x.
- It is a continuous function of $p(x)$.
- It is additive for independent events.

This last property is easy to prove:

$$P(X = x, Y = y) = P_{XY}(x, y) = P_X(x) P_Y(y)$$

when X, Y are independent. Then

$$\gamma(x, y) = -\log P_{XY}(x, y) = \gamma(x) + \gamma(y).$$

Definition 1.5. We can now define the *Shannon entropy* of X to be

$$H(X) \equiv \mathbb{E}(\gamma(X)) = \sum_{x \in J} (-\log p(x)) p(x), \tag{1.6}$$

the expected value of the surprisal. We see again that $H(X)$ does not depend on the actual outcomes themselves but only on the probability distribution $P(X)$.

As a matter of convention we will take logs to be $\log \equiv \log_2$, and for events which are impossible, $P(x) = 0$, we have $0 \log 0 = 0$ (which one can prove by taking the limit $\lim_{u \to 0} u \log u = 0$).

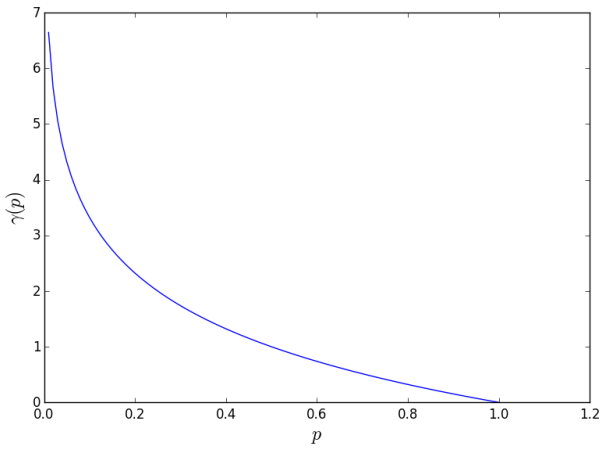


FIGURE 2. The surprisal $\gamma(p) \equiv -\log_2 p$ as a function of p, the probability of some event. Certainties ($p = 1$) are not very surprising, whereas very rare events ($p \ll 1$) are surprising, and so get $\gamma = 0$ and γ large respectively.

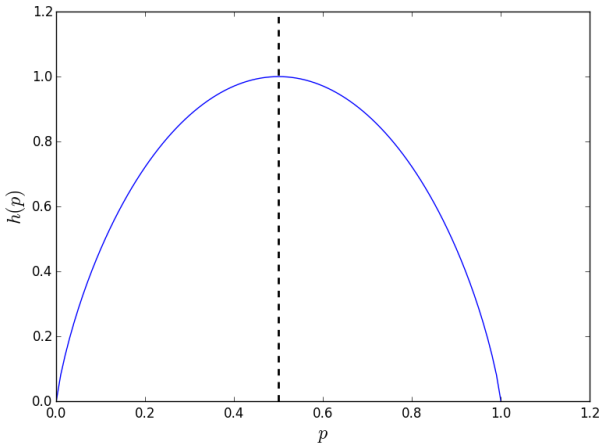


FIGURE 3. The Shannon entropy of a binary event where there are two possible outcomes, one of which happens with probability p and the other with $1 - p$. When $p = 0.5$, our ignorance is at a maximum– we know nothing a priori about what our generator will spit out.

Binary entropy Consider an event which has two possible outcomes, $X \sim P(x), x \in J = \{0, 1\}$ where $P(X = 0) = p$ and $P(X = 1) = 1 - p$. Then the Shannon entropy is

$$H(X) = -p \log p - (1 - p) \log(1 - p) \equiv h(p). \tag{1.7}$$

We see that if the probability is $p = 1/2$, then we have no information a priori about this systems– the entropy is maximized. $h(p)$ is a continuous function of p, and it is concave. See the illustration in Fig. **??**.

Definition 1.8. We can also define a different entropy, the Rényi entropy, which is

$$H_\alpha(X) = \frac{1}{1 - \alpha} \log \left(\sum_{x \in J} p(x)^\alpha \right), \tag{1.9}$$

with $\alpha \in (1,2]$. As an exercise, we can verify that $\lim_{\alpha\to 1} H_\alpha(X) = H(X)$, i.e. the Renyi entropy reduces to the Shannon entropy.[2]

Why do we choose to work with the Shannon entropy? It has to do with the operational interpretation– the Shannon entropy represents an optimal rate of data compression, i.e. the data compression limit.

In CIT, a classical information source emits some messages/data/signals/information. For instance, J could output a binary output or perhaps telegraph English (26 letters and a space). Now, the simplest class of sources is *memoryless*– they are "independent identically distributed" sources (i.i.d.), which means that successive messages are independent of each other, and they are identically distributed.

Definition 1.10. Suppose we have some random variables $U_1, U_2, \ldots, U_n$ with $U_i \sim p(u), u \in J$. We say these are *identically distributed* if

$$p(u) = P(U_k = u), u \in J \quad \forall 1 \leq k \leq n.$$

We could study a signal emitted by n uses of the source to get some sequence $\underline{u}^{(n)} = (u_1, u_2, \ldots, u_n)$.

Definition 1.11. Moreover, if the probability mass function takes the form

$$\begin{aligned} p(\underline{u}^{(n)}) &= P(U_1, \ldots, U_n = u_n) \\ &= p(u_1) \ldots p(u_n). \end{aligned}$$

If the source is indeed independent and identically distributed, then it makes sense to describe it by a sincle probability mass function, $U \sim p(u)$, so that the Shannon entropy of the source can be said to be

$$H(U) = -\sum_{u\in J} p(u) \log p(u). \tag{1.12}$$

Another guiding question. Why is data compression possible? Our information source has some *redundancy*. For instance, in the English language, certain letters are more common than others, so we can encode something that is more common in a shorter string in anticipation it will be used more often.

This sort of scheme is known as variable length coding, e.g. we might encode the letter "e" as the string 10 and the letter "z" as 11000. In contrast, we could also use a fixed length coding scheme where we have a "typical set", a subset of our total outcomes J^n (things we might like to encode). Our typical set then has a one-to-one mapping to the set of encoded messages, e.g. $\{0,1\}^m$, so we can always recover them precisely, while several outcomes outside the typical set might map to the same encoded message. There's some probability that we'll want to encode things outside the typical set, and in decoding we'll get the original message a little bit wrong. But if we choose the typical set well, this can be made to be a rare occurrence. We are usually interested in *asymptotic i.i.d.* settings, i.e. in the limit as the size of the set of possible messages to be encoded goes to ∞.

Example 1.13. Suppose we have a horse race with eight horses. They have labels $1, 2 \ldots, 8$, and the message we would like to encode is the label of the winning horse. A priori, we only need 3 bits to encode the label since 2^n different messages can be stored in n bits.

However, what if the horses are not all equally fast (i.e. likely to win)? Suppose that p_i is the probability of the ith horse winning, such that

$$p_i = 1/2, 1/4, 1/8, 1/16, 1/64, \ldots, 1/64.$$

[2]The proof is fairly quick. First note that as $\alpha \to 1$, the denoninator $1-\alpha$ goes to zero and the log becomes $\log(\sum_{x\in J} p(x)) = \log 1 = 0$, so we can apply L'Hôpital's rule and take some derivatives. Note also that $\frac{d}{dx} a^x = \frac{d}{dx} e^{\log a^x} = \frac{d}{dx} e^{x \log a} = \log a e^{x \log a} = a^x \log a$. Thus by L'Hôpital's rule,

$$\begin{aligned} \lim_{\alpha\to 1} H_\alpha(X) &= \lim_{\alpha\to 1} \frac{1}{1-\alpha} \log\left(\sum_{x\in J} p(x)^\alpha\right) \\ &= \lim_{\alpha\to 1} \frac{1}{(-1)} \frac{\sum_{x\in J}(p(x)^\alpha \log p(x))}{\sum_{x\in J} p(x)^\alpha} \\ &= -p(x) \log p(x) = H(X). \end{aligned}$$

Technically I have done this calculation with a natural log rather than a base 2 log, but the result is the same, since the numerical factor from taking the derivative of the log cancels with the factor from rewriting the derivative of $p(x)^a$ in terms of a base 2 log. ☒

Now we assign the following code words:

$$\begin{aligned} C(1) &= 0 \\ C(2) &= 10 \\ C(3) &= 110 \\ C(4) &= 1110 \\ C(5) &= 111100 \\ C(6) &= 111101 \\ C(7) &= 111110 \\ C(8) &= 111111. \end{aligned}$$

Let l_i be the length of the ith codeword, e.g. $l_5 = 6$. We can compute that the average length of a code is then $\sum p_i l_i = 2$, and we've chosen a "prefix-free code" so that a sequence like 10011001110 can be uniquely decoded to a sequence of winners from our code words. That is, no codeword is a prefix of any other code.[3]

Let's compute the expected length of the codeword– it is

$$\sum_i p_i l_i = 1 \times \frac{1}{2} + 2 \times \frac{1}{4} + 3 \times \frac{1}{8} + 4 \times \frac{1}{16} + 4 \times \frac{1}{64} \times 6 = 2, \tag{1.14}$$

and this is exactly the Shannon entropy of the system, as expected.

Lecture 2.

Monday, January 21, 2019

Last time, we introduced Shannon's Source Coding Theorem:

Theorem 2.1. *For an i.i.d (memoryless) source, the optimal rate of reliable data compression (i.e. the data compression limit) is precisely the Shannon entropy $H(X)$ of the source.*

We started by saying that if we have an iid source, we can model it by a collection of n sources $U_1, U_2 \ldots, U_n$ which outputs a length-n vector $\underline{u}^{(n)} = (u_1, \ldots, u_n) u_i \in J$. For an iid source, all the sources have the same probability mass function,

$$U_i \sim p(u), u \in J,$$

which means that we can equivalently model the source as a single source,

$$U \sim p(u), u \in J; p(\underline{u}^{(n)}) = \prod_{i=1}^{n} P(U_i = u_i) = p(u_1) \ldots p(u_n).$$

The Shannon entropy of the source is given as usal by

$$H(U) = -\sum_{u \in J} p(u) \log p(u). \tag{2.2}$$

Now let us define a compression map.

Definition 2.3. A *compression map* of *rate* R is a map $\mathcal{C}$ with

$$\mathcal{C}^n : \underline{u}^{(n)} = (u_1, \ldots, u_n) \mapsto \underline{x}^{m_n} = (x_1, \ldots, x_{m_n}) \in \{0,1\}^{m_n}. \tag{2.4}$$

That is, $\mathcal{C}$ maps our output string of length n to a compressed (encoded) string $\underline{x}$ of length m_n. We say that the *rate* of encoding is then

$$R = \frac{m_n}{n} = \frac{\text{number of bits in codeword}}{\text{number of uses of source}}. \tag{2.5}$$

If a compression scheme has rate R, then we assign unique codewords to $2^{\lceil nR \rceil}$ messages.

[3] For the sequence 10011001110, we know that the first winner was the horse corresponding to 10, horse 2. The next winner was horse 1 with code 0. This sequence breaks up as 10|0|110|0|1110, so the winners were 2, 1, 3, 1, and 4 in that order.

Question: when is such a map $\mathcal{C}^n$ a compression map? If our source outputs n values in the alphabet J, then we have total possibilities

$$|J|^n = 2^{n \log |J|}. \tag{2.6}$$

These can be stored in $n \log |J|$ bits. Thus c^n is a compression map if $m_n < n \log |J|$, i.e. if we encode the output in fewer bits than it would take to uniquely encode every single string in the naive binary way.

We can of course also define a decompression map:

Definition 2.7. A decompression map $\mathcal{D}^n$ is a map

$$\mathcal{D}^n : \underline{x}^{m_n} \in \{0,1\}^{m_n} \mapsto \underline{u}^{(n)} = (u_1, \ldots, u_n), \tag{2.8}$$

i.e. which takes us back to the original length-n strings of source outputs.

Now we can ask what the probability of a successful encoding and decoding is– namely,

$$\sum_{\underline{u}^{(n)} \in J^n} p(\underline{u}^{(n)}) P\Big(\mathcal{D}^n(\mathcal{C}^n(\underline{u}^{(n)})) \neq \underline{u}^{(n)}\Big) \tag{2.9}$$

is the average probability of error of the compression process. We write this as $P_{av}^{(n)}(C_n)$, where C_n denotes an encoding and decoding scheme.

Definition 2.10. C_n is a triple defined to be $C_n \equiv (\mathcal{C}^n, \mathcal{D}^n, R)$ which represents a choice of code. We say that a code is *reliable* if $P_{av}^{(n)} \to 0$ in the limit as $n \to \infty$. That is, $\forall \epsilon \in (0,1), \exists n$ such that $p_{av}^{(n)} \leq \epsilon$.

Then there is an optimal rate of data compression,

$$R_\infty = \inf\{R : \exists C_n(\mathcal{C}^n, \mathcal{D}^n, R) \text{ s.t. } p_{av}^{(n)}(C_n) \to 0 \text{ as } n \to \infty\}. \tag{2.11}$$

That is, R_∞ is effectively the minimum rate R of all reliable coding schemes. What Shannon's source coding theorem tells us is that $R_\infty = H(U)$. The lowest rate (highest density, if you like) we can reliably compress an iid source to is given by the Shannon entropy.

Definition 2.12. An ϵ-typical sequence is a sequence defined as follows. Fix $\epsilon \in (0,1)$ and take an iid source with $U \sim p(u), u \in J$ which gives us a length-n output $\underline{u}^{(n)} = (u_1, \ldots, u_n)$. Then if

$$2^{-n(H(U)+\epsilon)} \leq p(\underline{u}^{(n)}) \leq 2^{-n(H(U)-\epsilon)}, \tag{2.13}$$

we say that $\underline{u}^{(n)}$ is an ϵ-typical sequence.

Definition 2.14. An *ϵ-typical set* is then defined to be the set

$$T_\epsilon^{(n)} = \{\underline{u}^{(n)} \in J^n \text{ such that } \textbf{??} \text{ holds}\}. \tag{2.15}$$

In the asymptotic limit let us observe that

$$p(\underline{u}^{(n)}) \approx 2^{-nH(U)}, \tag{2.16}$$

so all ϵ-typical sequences are almost equiprobable since ϵ can be made arbitrarily small. Does this agree with our intuitive notion of a typical sequence? Yes– take a sequence $\underline{u}^{(n)} = (u_1, \ldots, u_n), u_i \in J$. Note that for every $u \in J$, the number of times we expect to u to appear in a string $\underline{u}^{(n)}$ is simply $np(u)$.

Our intuition tells us that any typical sequence should therefore fit this expectation.[4] The probability of getting one specific typical sequence is

$$\begin{aligned} p(\underline{u}^{(n)}) &\simeq \prod_{u\in J} p(u)^{np(u)} \\ &= \prod_{u} 2^{np(u)\log p(u)} \\ &= 2^{n\sum p(u)\log p(u)} \\ &= 2^{-nH(U)}. \end{aligned}$$

So this agrees well with our formal definition of a typical sequence. Note that there is a difference between typical and high-probability– we'll investigate this distinction further on the example sheet.

Now, typical sequences have some nice properties.

Theorem 2.17 (Typical sequence theorem)**.** *$\forall \delta > 0$ and large n,*

- $H(U) - \epsilon \leq -\frac{1}{n}\log p(\underline{u}^{(n)}) \leq H(u) + \epsilon$[5]
- $P(T_\epsilon^{(n)}) := \sum_{\underline{u}^{(n)}\in T_\epsilon^{(n)}} p(\underline{u}^{(n)}) > 1-\delta$. *That is, the probability of getting any typical sequence (as a subset of possible outputs) can be made arbitrarily close to* 1.
- $2^{n(H(U)-\epsilon)}(1-\delta) < |T_\epsilon^{(n)}| \leq 2^{n(H(U)+\epsilon)}$, *where $|T_\epsilon^{(n)}|$ is the number of typical (length n) sequences.*[6]

Since $\epsilon > 0$, we see that in the limit $\epsilon \to 0$,

$$|T_\epsilon^{(n)}| \to 2^{nH(U)}. \tag{2.18}$$

That is, we need $nH(U)$ bits to store all the typical sequences.

Now we can state Shannon's theorem formally.

Theorem 2.19 (Shannon's Source Coding Theorem)**.** *Suppose we have an iid source U with Shannon entropy $H(U)$.*

(a) *(Achievability) Suppose $R > H(U)$. Then $\exists$ a reliable compression-decompression scheme of rate R.*
(b) *(Converse) For $R < H(U)$, any compression-decompression scheme is not reliable.*

Constructive proof of (a) Let us suppose that $R > H(U)$. We fix $\epsilon \in (0,1)$ such that $R > H(U) + \epsilon$ (for instance, $\epsilon = (R - H(U))/2$). Then we choose n large enough (i.e. the asymptotic limit) such that $T_\epsilon^{(n)}$ satisfies the conditions of the typical sequence theorem. Then we can write

$$|T_\epsilon^{(n)}| \leq 2^{n(H(U)+\epsilon)} < 2^{nR}. \tag{2.20}$$

---

[4] To make this more concrete, suppose we have a weighted coin. The weighted coin has outcomes h and t (heads and tails), and it produces h with probability $3/4$ and t with probability $1/4$. If we flip the coin n times, we expect to see about $n \times p(h) = n \times 3/4$ heads and $n \times p(t) = n \times 1/4$ tails since each flip is independent. If $n = 4$, for instance, then a "typical sequence" will have three heads and one tails.

Consider a specific example of a length-4 typical sequence, *hhht* in that order. The probability of getting this specific sequence is $p(h) \times p(h) \times p(h) \times p(t) = 27/256$. We could have written this as $(p(h))^{np(h)} \times (p(t))^{np(t)}$, or equivalently $2^{np(h)\times\log p(h)} \times 2^{np(t)\times\log p(t)}$. Combining terms, we see that this is just $2^{n(p(h)\log p(h)+p(t)\log p(t))} = 2^{-nH(U)}$.

This is not the probability of getting *any* sequence which fits the typical sequence condition! That probability would be something like $\binom{4}{3}$ times the probability we got, since we want exactly three heads. However, we will put a bound on this quantity shortly.

[5] This follows from taking the log of the definition of an ϵ-typical sequence and dividing by $-n$.

[6] Since the probability of any individual typical sequence is bounded from below by definition and there are $|T_\epsilon^{(n)}|$ such sequences, the probability of getting *any* typical sequence is bounded by

$$2^{-n(H(U)+\epsilon)}|T_\epsilon^{(n)}| \leq \sum_{\underline{u}^{(n)}\in T_\epsilon^{(n)}} p(\underline{u}^{(n)}) \leq 1.$$

This leads us to conclude that $|T_\epsilon^{(n)}| \leq 2^{n(H(U)+\epsilon)}$.

However, $|T_\epsilon^{(n)}|$ is also bounded from below. We know from the previous property and the definition of a typical sequence that

$$1-\delta < \sum p(\underline{u}^{(n)}) \leq 2^{-n(H(U)-\epsilon)}|T_\epsilon^{(n)}|,$$

so $2^{n(H(U)-\epsilon)}(1-\delta) < |T_\epsilon^{(n)}|$.

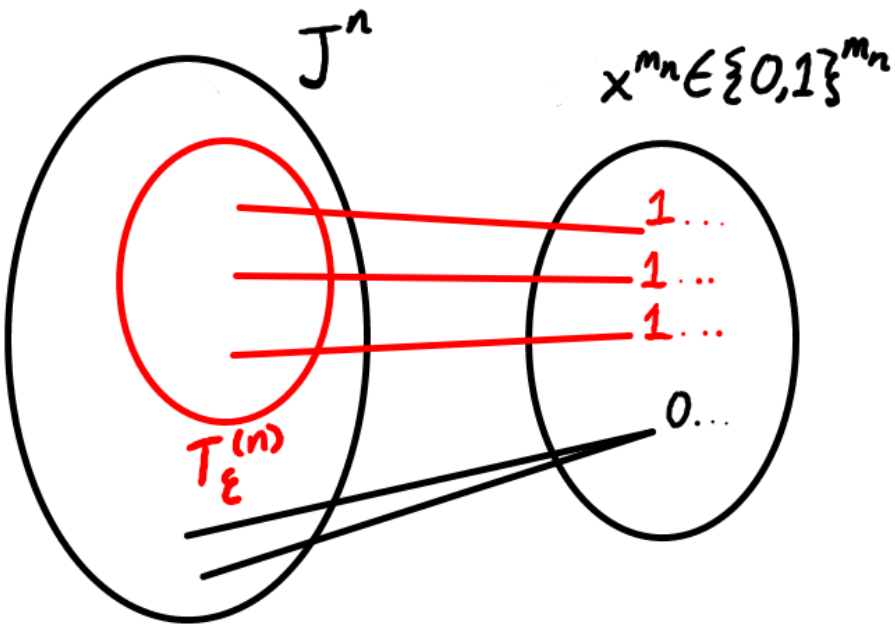


FIGURE 4. An illustration of the encoding procedure for the achievability part of the Shannon source coding theorem. Of our source's possible outputs J^n, we set up a one-to-one encoding of the typical set $T_\epsilon^{(n)}$, (red ellipse), and send all other elements of J^n to some random value in our set of codewords.

Now we divide our set of sequences J^n into the typical set T_ϵ^n and its complement $A_\epsilon^n = J^n \setminus T_\epsilon^n$. Let us then order the elements of our typical set, i.e. we assign some labels/indices to all the elements. Since $|T_\epsilon^n| < 2^{nR}$, we need at most nR bits to store all the labels of the typical sequences (i.e. the ones we always want to recover reliably).[7]

With our encoding scheme for the typical set in hand, let us preface our encoding with a 1, i.e. a *flag bit*. So the typical set elements will be encoded as

$$\underline{u}^{(n)} \in T_\epsilon^n \mapsto 1\underbrace{011\ldots1}_{\lceil nR \rceil}. \tag{2.21}$$

Our codewords will be of length $\lceil nR \rceil + 1$, and we can assign the complement A_ϵ^n to some random codeword beginning with a 0 instead. This procedure is shown in Fig. **??**. So our rate of success when we decode will not be exactly 1– we can perfectly decode typical set elements, but there is some loss when we encode elements outside the typical set. However, things are not so bad. Let us take the limit as $n \to \infty$ and look at the failure probability $p_{av}^{(n)}$.

$$\begin{aligned} p_{av}^{(n)} &:= \sum p(\underline{u}^{(n)}) P\Big(\mathcal{D}^n(\mathcal{C}^n(\underline{u}^{(n)})) \neq \underline{u}^{(n)}\Big) \\ &= \sum_{\underline{u}^{(n)} \in T_\epsilon^n} p(\underline{u}^{(n)}) P(\underline{u}'^{(n)} \neq \underline{u}^{(n)}) + \sum_{\underline{u}^{(n)} \in A_\epsilon^n} p(\underline{u}^{(n)}) P(\underline{u}'^{(n)} \neq \underline{u}^{(n)}). \end{aligned}$$

But the first term is zero since we can always decode typical set elements, and the second part can be made to be arbitrarily small ($< \delta$) by the typical sequence theorem. Therefore we conclude that our scheme is reliable.[8] ⊠

[7] As nR may not be an integer, we'll practically need at most $\lceil nR \rceil$ bits.

[8] That is, since $P(T_\epsilon^n) > 1 - \delta$, it follows that $P(A_\epsilon^n) < \delta$ in the large-n limit. So the nonzero failure rate is washed out by the fact that

$$\sum_{\underline{u}^{(n)} \in A_\epsilon^n} p(\underline{u}^{(n)}) P(\underline{u}'^{(n)} \neq \underline{u}^{(n)}) \leq \sum_{\underline{u}^{(n)} \in A_\epsilon^n} p(\underline{u}^{(n)}) = P(A_\epsilon^n) < \delta$$

Lemma 2.22. *Suppose we have a set S^n which has size $|S^n| = 2^{nR}$, with $R < H(U)$. $\forall \delta > 0, S_n \subset J^n$ s.t. $|S^n| = 2^{nR}$ with $R < H(U)$, we have $P(S^n) < \delta$ for n large enough.*

This implies the converse, and is in the course notes (but is useful to think on by oneself).

Non-lectured aside: the converse I'll present here an argument for the above lemma. A similar exposition appears in the official course notes.

We have some set S^n with size $|S^n| = 2^{nR}$. That is, we can encode and decode at most 2^{nR} elements with perfect precision. What elements should we choose?

We know that the probability of our source producing any element in the atypical set $A_\epsilon^{(n)}$ becomes arbitrarily small by the typical sequence theorem, so in order to give our encoding scheme the best chance of success, we should not bother with encoding any elements in $A_\epsilon^{(n)}$. But note that

$$|S^n| = 2^{nR} < 2^{nH(U)} < |T_\epsilon^{(n)}|$$

for some $\epsilon > 0$, so we cannot encode the entire typical set. At best, we can encode a subset of $T_\epsilon^{(n)}$.

Let's do that, then. We take $S^n \subset T_\epsilon^{(n)}$, and note that the probability of any individual typical sequence is $2^{-nH(U)}$. Since we have 2^{nR} such sequences in S^n, the probability of our source producing any sequence in S^n is simply

$$P(S^n) = \sum_{\underline{u}^{(n)} \in S^n} p(\underline{u}^{(n)}) = 2^{nR} 2^{-nH(U)} = 2^{-n(H(U)-R)}. \tag{2.23}$$

Since $R < H(U)$ by assumption, $H(U) - R > 0 \implies P(S^n) = 2^{-n(H(U)-R)} \to 0$ as $n \to \infty$. Thus $\forall \delta > 0$, $\exists N$ such that $P(S^n) < \delta$ for $n \geq N$.

One interpretation of this is as follows– we tried to encode a subset of the typical set, hoping that any elements in $T_\epsilon^{(n)} \setminus S^n$ wouldn't totally ruin our encoding scheme. However, what we didn't account for was the limit $n \to \infty$. The number of typical sequences grows too fast for our encoding scheme to keep up, so that the probability of our source producing a typical sequence we didn't encode is

$$P(T_\epsilon^n) - P(S^n) > 1 - \delta - 2^{-n(H(U)-R)}, \tag{2.24}$$

which can be made arbitrarily close to 1. The moral of the story is that if we don't encode the entire typical set at a minimum, our scheme is doomed to fail.

Lecture 3.

Wednesday, January 23, 2019

Let's recall the statement of Shannon's source coding theorem. Shannon tells us that if we have an iid source $U \sim p(u); u \in J$ with Shannon entropy $H(U)$, then there is a fundamental limit on data compression given by $H(U)$ such that for any rate $R > H(U)$, there exists a reliable compression-decompression scheme of rate R, and conversely for any rate $R < H(U)$, any scheme of rate R will not be reliable.

See my notes from last lecture for a heuristic argument of the converse. The formal argument can be made with ϵs and δs– for example, my statement that we need not consider elements in $A_\epsilon^{(n)}$ is equivalent to $\sum_{\underline{u}^{(n)} \in S^n \cap A_\epsilon^n} p(\underline{u}^{(n)}) \leq P(A_\epsilon^n \to 0$.

Entropies Consider a pair of random variables X, Y with *joint probability*

$$P(X = x, Y = y) = P_{XY}(x, y) = p(x, y). \tag{3.1}$$

Here, $x \in J_X$ some alphabet and similarly $y \in J_Y$. We can also define the conditional probability

$$P(Y = y | X = x) = p(y|x), \tag{3.2}$$

the probability of y given x.

Definition 3.3. Now we have the *joint entropy*, which is

$$H(X, Y) \equiv - \sum_{x \in J_X, y \in J_Y} p(x, y) \log p(x, y). \tag{3.4}$$

for δ arbitrarily small.

Definition 3.5. We also have the *conditional entropy*, which is

$$\begin{aligned} H(Y|X) &\equiv \sum_x p(x) H(Y|X=x) \\ &= -\sum_x p(x) \sum_y p(y|x) \log p(y|x). \end{aligned}$$

But we can simplify this to write

$$H(Y|X) = -\sum p(x,y) \log p(y|x), \tag{3.6}$$

which implies that

$$p(x,y) = p(x)p(y|x) = p(y)p(x|y). \tag{3.7}$$

This leads us to a chain rule,

$$H(X,Y) = H(Y|X) + H(X). \tag{3.8}$$

We also have the notion of a relative entropy, which measures a "distance" between two probability distributions. Suppose we have distributions $p = \{p(x)\}_{x\in J}$ and $q = \{q(x)\}_x \in J$, Let us assume that the supp$p \subseteq$ supp$q$, with supp$p = \{x \in J : p(x) > 0\}$. This implies that $q(x) = 0 \implies p(x) = 0$, which we denote $p \ll q$.

Definition 3.9. Thus we define the *relative entropy* to be

$$D(p||q) \equiv \sum_{x\in J} p(x) \log \frac{p(x)}{q(x)}. \tag{3.10}$$

If $p \ll q$, then this is well-defined (otherwise we might have $q \to 0$ with p nonzero). Taking $0 \log \frac{0}{q(x)} = 0$ we see that this represents a sort of distance,

$$D(p||q) \geq 0 \tag{3.11}$$

with equality iff $p = q$.

This is not quite a true metric, since it is not symmetric, $D(p||q) \neq D(q||p)$, and moreover it does not satisfy a triangle inequality, i.e. $D(p||r) \not\leq D(p||q) + D(q||r)$.

Using the relative entropy, we can now define a useful quantity known as the mutual information.

Definition 3.12. The mutual information between two sources X and Y is

$$\begin{aligned} I(X;Y) &= H(X) + H(Y) - H(X,Y) \\ &= H(X) - H(X|Y). \end{aligned}$$

The mutual information has some intuitive properties.

- $I(X:X) = H(X)$, since $I(X;X) = H(X) + H(X) - H(X,X) = H(X)$.
- $I(X;Y) = I(Y;X)$
- if X, Y independent, then $I(X;Y) = 0$.

Suppose now we have P, Q taking non-negative real values, with $Q(x) = 0 \implies P(X) = 0$. THus the relative entropy is

$$D(P||Q) = \sum P(x) \log \frac{P(x)}{Q(x)}.$$

What if $P(x) = p(x), x \in J$ and $Q(x) = 1 \forall x \in J$? Then

$$D(P||Q) = \sum_x p(x) \log p(x) = -H(X). \tag{3.13}$$

It's almost trivial to check that if $Q(x) = \frac{1}{|J|}$ instead, then we would get an additional factor of $-\log |J|$.

Exercise 3.14. Check that the mutual information satisfies

$$I(X;Y) = D(p(x,y)||p(x)p(y)). \tag{3.15}$$

Let's take a minute to prove the non-negativity of the relative entropy. That is, $D(p||q) \geq 0$.

Proof. By definition,

$$D(p||q) = \sum_{x \in J} p(x) \log \frac{p(x)}{q(x)}. \tag{3.16}$$

Let us define a set A such that

$$A = \{x \in J \text{ s.t. } p(x) >)\}.$$

Thus A is the support of J. We can compute

$$-D(p||q) = \sum p(x) \log \frac{q(x)}{p(x)} \tag{3.17}$$

$$= \mathbb{E}_p \left(\log \frac{q(X)}{p(X)} \right). \tag{3.18}$$

Note that Xs denote random variables, while xs indicate the values they take.

Jensen's inequality from probability theory tells us that for convave functions f, $\mathbb{E}(f(X)) \leq f(\mathbb{E}(X))$.

We conclude that

$$\begin{aligned} -D(p||Q) &\leq \log(\mathbb{E}_p \frac{q(X)}{p(X)}) \\ &= \log \sum_{x \in A} p(x) \frac{q(x)}{p(x)} \\ &\leq \log \sum_{x} \in J q(x) \\ &= \log 1 = 0 \\ &\implies D(p||Q) \geq 0. \end{aligned}$$

⊠

Suppose we had a distribution $p = \{p(x)\}, q(x) \frac{1}{|J|} \forall x \in J$ as before. Then

$$0 \leq D(p||q) = \sum p(x) \log \frac{p(x)}{(1/|J|)} \tag{3.19}$$

$$= -H(X) + \sum p(x) \log |J| \tag{3.20}$$

$$\implies H(X) \leq \log |J|. \tag{3.21}$$

Lecture 4.

Friday, January 25, 2019

Last time, we introduced many important classical concepts. We talked about the mutual (common) information $I(x : Y)$ between two sources, arguing that

$$\begin{aligned} I(X : Y) &= H(X) + H(Y) - H(X, Y) \\ &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X). \end{aligned}$$

In particular we find that $I(X : X) = H(X), I(X : Y) = I(Y : X)$, and $I(X : Y) = 0$ iff X, Y are independent.

We can also prove that the mutual information is non-negative,

$$I(X : Y) \geq 0, \tag{4.1}$$

which follows from writing in terms of the conditional entropy as $H(X) - H(X|Y) \geq 0$. Equivalently we should show that

$$H(X|Y) \leq H(X). \tag{4.2}$$

That is, *conditioning reduces entropy*.

We may describe the concavity of $H(X)$– that is, for two sources with $X, Y; J$ with $\lambda \in [0, 1]$

$$H(\lambda p_X + (1 - \lambda) p_Y) \geq \lambda H(p_x) + (1 - \lambda) H(p_Y), \tag{4.3}$$

which we will prove on the first examples sheet. This is analogous to what we showed in a few lines about the binary entropy.

The Shannon entropy of $H(X,Y)$ (where we simply replace $p(x)$s in the definition of $H(X)$ with $p(x,y)$) is constrained by the following inequality:

$$H(X,Y) \geq H(X) + H(Y). \tag{4.4}$$

This property is known as *subadditivity*.

We also have the property that the conditional entropy is non-negative–

$$H(X|Y) \geq 0. \tag{4.5}$$

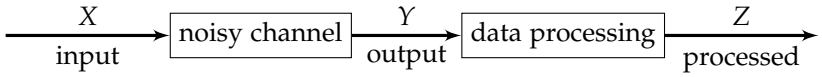
Equivalently, $H(X,Y) - H(Y) \geq 0$ We shall see that once we introduce quantum correlations, this will no longer be true.

Data processing inequality Suppose we have some variables $X_1, X_2, \ldots$. In a Markov chain, we say that the probability of some outcome $X_n = x_n$ in a chain is

$$P(X_n = x_n | X_1 = x_1 \ldots X_{n-1} = x_{n-1}) = P(X_n = x_n | X_{n-1} = x_{n-1}). \tag{4.6}$$

That is, the value of a Markov chain at a position n depends only on its value at $n-1$.

Consider a simple Markov chain with three variables, $X \to Y \to Z$.



Then by the definition of a Markov chain, $P(Z = z|X = x, Y = y) = P(Z = z|Y = y)$, and we can prove that

$$I(X:Z) \leq I(X:Y), \tag{4.7}$$

known as the *data processing inequality* (DPI). That is, there is no data processing that can increase the correlation between two random variables.

Chain rules Chain rules are relations between different entropy quantities, e.g. $H(X,Y) = H(X) + H(Y|X)$. Suppose we have three random variables X, Y, Z with a joint probability of $p(x,y,z)$.

Exercise 4.8. Prove that

$$H(X,Y,Z) = H(X) + H(Y|X) + H(Z|X,Y). \tag{4.9}$$

Definition 4.10. Now one can define the *conditional mutual information* as follows:

$$I(X:Y|Z) := H(X|Z) - H(X|Y,Z) \geq 0, \tag{4.11}$$

with equality when $X - Y - Z$ forms a Markov chain.

We have one more topic for classical information theory– it is *Shannon's Noisy Channel Coding Theorem.* As usual, let us work in the asymptotic iid limit.

Suppose we have some source X producing outputs in an alphabet J_X, and some received signals $Y \in J_Y$. We also have a noisy channel $\mathcal{N} : J_X \to J_Y$, and a stochastic map, defined to be a set of probabilities $\{p(y|x)\}$.

Here's the setup. Alice wants to send a message m to her friend Bob. To do this, she takes her message $m \in \mathcal{M}$ a set of messages and runs an encoding process $\mathcal{E}_n$ to produce a codeword $x_m^{(n)}$. She uses the (possibly noisy) channel $\mathcal{N}$ multiple times, say n times, to send a transmitted message $y_m^{(n)} \neq x_m^{(n)}$, which Bob then runs a decoding process $\mathcal{D}_n$ on to get a final decoded message m'.

If $m' \neq m$, we have gotten an error. In the $n \to \infty$ limit, we would like the probability of error $p_e rr^{(n)} = p(m' \neq m) \to 0$.

In some sense, encoding is like the dual process of compression. In encoding, we add redundancy in a controlled manner to account for the potential noise of the channel $\mathcal{N}$.

Definition 4.12. We define a *discrete channel* to be the following:

- An input alphabet J_X
- An output alphabet J_Y
- A set of conditional probabilities (dependent on the number of uses n) $\{p(\underline{y}^n|\underline{x}^n)\}$.

The input to n uses of the channel sends n uses of the source, $\underline{x}^{(n)} = (x_1, \dots, x_n) \in J_X^n$ to an output $\underline{y}^{(n)} = (y_1 \dots y_n) \in J_Y^n$ with probability $p(\underline{y}^n | \underline{x}^n)$.

We can consider memoryless channels, i.e. where the probability of n uses completely separates into n independent uses of the channel as

$$p(\underline{y}^n | \underline{x}^n) = \prod_{i=1}^{n} p(y_i | x_i). \tag{4.13}$$

For a memoryless channel, we may write the transition probabilities as a *channel matrix*,

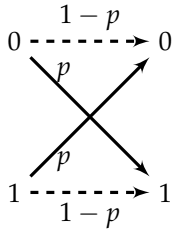
$$\begin{bmatrix} p_{11} & \cdots & p_{1|J_Y|} \\ \vdots & & \vdots \\ p_{|J_X|1} & \cdots & p_{|J_X||J_Y|} \end{bmatrix}. \tag{4.14}$$

If the rows are permutations, then the channel matrix is symmetric.

Example 4.15. Consider a memoryless binary symmetric channel (m.b.s.c). Thus the set of inputs and the set of outputs are $J_X = J_Y = \{0, 1\}$. If the channel sends $0 \mapsto 1$ with probability p and $0 \mapsto 0$ with probability $1 - p$ (that is, $p(0|1) = p$), then the channel matrix takes the form

$$\begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}, \tag{4.16}$$

which we can represent in the following diagram (with initial states on the left and final states on the right).



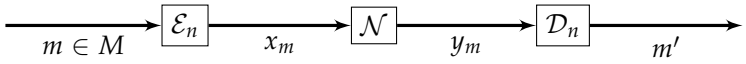
Now consider the following encoding scheme. We encode $0 \to 000$ and $1 \to 111$. Suppose we got 010 as the output. What do we think the input was?

Probably 0, since it could have come from 000 with the middle bit flipped. But it could have come from 111 with the first and last bits flipped, too.

Now, a simple exercise. For what p is this encoding-decoding scheme better than just sending the original message? Intuitively, we might guess $p = 1/2$, and this is correct. But we should prove it.

Moreover, what is the correspondence between the input and output of the channel? We see that it's certainly not one-to-one, from the last example. So we might have to guess what the original message was. However, what Shannon realized was that for certain elements of J_X^n, their images under the noisy channel map will be disjoint, so these elements will make very good codewords since we can always decode the output even after noise is introduced– see Fig. **??**.

We won't do the full proof of the theorem today, but we can introduce the setup. Suppose Alice has a message $[M] = \{1, 2, \dots, M\}$ she would like to send to Bob. She has a noisy channel $\mathcal{N} : J_X \to J_Y$ with some transition probabilities $p(\underline{y}^n | \underline{x}^n)$.



(a) First, Alice can choose an encoding scheme $\mathcal{E}_n : [M] \to J_X^n$ where $\forall m \in [M], \mathcal{E}_n(m) = \underline{x}^{(n)} \in J_X^n$.
(b) She then sends her message through the channel $\mathcal{N}^{(n)} : x^{(n)} \to y^{(n)}$, producing some transmitted messages $y^{(n)}$ with some given probabilities.
(c) Bob receives the message and performs the decoding with $\mathcal{D}_n$ to get some decoded message $\mathcal{D}_n(\mathcal{N}^{(n)}(\mathcal{E}_n(M))) = m'$.

Thus the *maximum probability of error* is

$$\max m \in [M] P(\mathcal{D}_n(\mathcal{N}^{(n)}(\mathcal{E}_n(M))) \neq m) = p(\mathcal{E}_n, \mathcal{D}_n). \tag{4.17}$$

FIGURE 5. In a noisy channel, it might be the case that multiple inputs map to the same output, as in the left set of ovals. Both $\underline{x}^n$ and $\underline{x}^{n\prime}$ have been mapped to the same $\underline{y}^n$ with some probability. However, Shannon tells us that certain codewords will be transmitted as disjoint regions (red ovals) after being sent through the channel, so those codewords can be reliably decoded after transmission.

We say that the *rate* is the number of the bits of the message transmitted per use of the channel. That is,

$$R = \frac{\log M}{n} \tag{4.18}$$

since $M \approx 2^{\lfloor nR \rfloor}$.

Definition 4.19. We say that a rate is R is *achievable* if there exists a sequence $(\mathcal{E}_n, \mathcal{D}_n)$ with $M = 2^{nR}$ such that

$$\lim_{n\to\infty} p(\mathcal{E}_n, \mathcal{D}_n) = 0, \tag{4.20}$$

i.e. the maximum probability of error tends to zero as n goes to ∞.

We make one final definition for today.

Definition 4.21. The *channel capacity* is defined to be

$$C(\mathcal{N}) = \sup\{R : R \text{ is an achievable rate}\}, \tag{4.22}$$

the maximum achievable rate for a channel.

Non-lectured: m.b.s.c encoding For Example **??**, we were asked to consider a binary channel N with error probability p. That is, if we give it an input $x \in \{0,1\}$, we get an output $N(x) = y \in \{0,1\}$ such that $p(N(x) \neq x) = p$.

We came up with the following encoding scheme: send $0 \mapsto 000$ and $1 \mapsto 111$. To decode, we simply take a majority vote, e.g. 010 was "probably" 000, so the original message was 0. Now how much better can we do with this redundancy? Let's consider the possible inputs, how they would be encoded, and how often they would be correct.

Suppose we want to send $0 \mapsto 000$.

- With probability $(1-p)^3$, none of the three bits are flipped and we get 000 as the output. The process succeeds.
- With probability $3 \times p(1-p)^2$, exactly one of the three bits is flipped. (The factor of 3 comes from the fact that we could have flipped any of the three.) We still succeed.
- If two or three bits are flipped, we definitely fail.

By the symmetry of the problem, the success and failure probabilities are the same for $1 \mapsto 111$.

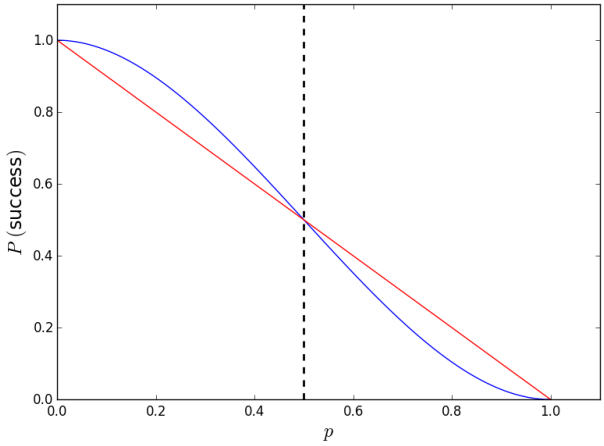
Let's add this up to get the total success probability:

$$(1-p)^3 + 3p(1-p)^2 = (1+2p)(1-p)^2. \tag{4.23}$$

When $p = 1/2$, the success probability of our scheme is

$$(1+2p)(1-p)^2 = (2)(1/2)^2 = 1/2. \tag{4.24}$$

We can nicely visualize this with the following graphic:



Here, the curved blue line is our three-bit scheme and the red line is the single-bit success probability $1-p$. For completeness, we can explicitly show that the crossover points occur when $P(\text{three bits}) - P(\text{one bit}) = (1+2p)(1-p)^2 - (1-p) = 0$. Rewriting, we have $(1-p)(1-2p)p = 0$, which clearly has zeroes at $p = 0, 1/2, 1$. If we now take a derivative, we see that $\frac{d}{dp}(P(\text{three bits}) - P(\text{one bit}))|_{p=1/2} = 1 - 6(1/2) + 6(1/2)^2 = -1/2$, so $P(\text{three bits}) > P(\text{one bit})$ for $p < 1/2$.