Information Theory

Delon Shen

Some notes I write while learning (classical and quantum) information theory. If you have any comments let me know at `hi@delonshen.com`.

Witten: Classical Information Theory **1**

Witten: Quantum Information Theory **8**

Wilde Chapter 2: Classical Shannon Theory **9**

Wilde Chapter 3: Noiseless Quantum Theory **19**

MacKay Chapter 1: Introduction to Information Theory **26**

Last updated on March 24, 2021

- Witten refers to Witten's *"A Mini-Introduction To Information Theory"*
- Wilde refers to Wilde's *"Quantum Information Theory"*
- MacKay refers to MacKay's *"Information Theory, Inference, and Learning Algorithms"*

Witten: Classical Information Theory

Started: February 01, 2021. Finished: February 06, 2021

Disaster! You've been struck by an acute and permanent case of locked in syndrome. In a tragic turn of events, your eyes begin to randomly move either up or down with probability p and $(1-p)$ every half a second. Even worse, you're on a conveyor belt that's slowly but steadily going towards a furnace (like that scene in Toy Story 3) and you know for certain that you'll be no more in 3 days. How inconvenient! For some reason you have some machines attached to you that records when you move your eyes up(denoted by H) and when you move your eyes down(denoted by A). Using this machine you can usually send messages to your loved ones that are of the form

$$HAHAHAHAHAHAHAHAAAHAAHAHAHAHAHHAA\ldots$$

But now it's just a random string generator. Given your deadline you guess that the resulting message will be N letters long. As you slowly inch towards your death you start to think about strange things. For example you think that when your loved ones look back on your final message that there will be around pN "H" characters and $(1-p)N$ "A" characters. How many messages with this combination of characters are there? Well from basic combinatorics we know that the number is

$$\frac{N!}{(pN)!((1-p)N)!} \approx \frac{N^N}{(pN)^{pN}((1-p)N)^{(1-p)N}} = \frac{1}{p^{p\times N}(1-p)^{(1-p)\times N}}$$

Now lets define a quantity called *Shannon entropy* S as

$$2^{NS} = \frac{1}{p^{p\times N}(1-p)^{(1-p)\times N}} \Rightarrow S = -p\log_2(p) - (1-p)\log_2(1-p)$$

Say we were sending the same kind of message that's N characters long but **we don't know the probabilities each letter would occur** then by combinatorics there would be 2^N possible messages. However we have more information than that fool, we know that H occurs with probability p and A occurs with probability $(1-p)$. That means we can send 2^{NS} messages. Or at least I think this is what we're saying. Just by knowing the probabilities each letter can occur has effectively (though not in reality) increased the length of our message by a factor of S! Each and every single letter in the string carries more information(or are maybe messages harder to decipher? I don't know yet). Lets extend this to the general case

Definition 1: (Shannon Entropy) Lets say we have a message that is composed of n different letters $\{a_1,\ldots,a_n\}$ where each letter occurs with probability $\{p_1,\ldots,p_n\}$. We define the Shannon Entropy S as

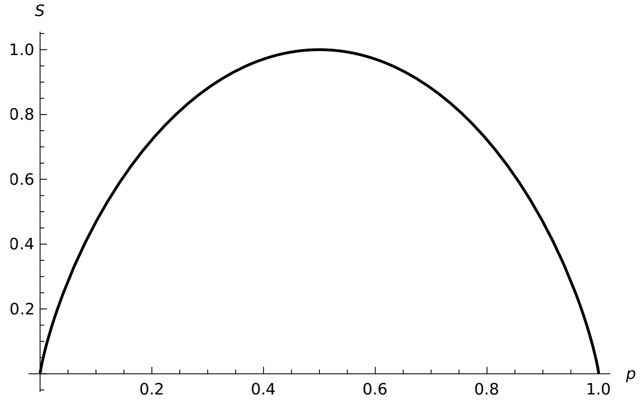
$$2^{NS}=\frac{N^N}{(p_1N)^{p_1N}\dots(p_nN)^{p_nN}}\Rightarrow S=-\sum_{i=1}^{N}p_i\log_2(p_i)$$

Lets notice a few things about Shannon entropy. First of all since the numbere of messages we can send has to be at least 1 we know that $S\geq 0$. Now lets also ask, what's the maximum possible entropy for an alphabet of k letters.

Example 1: (Maximizing Shannon Entropy) Before we consider a general alphabet of k letters lets just consider our simple two alphabet case. We had that

$$S=-p\log_2(p)-(1-p)\log_2(1-p)$$

Now lets plot the Shannon entropy versus p and see if we can't qualitatively guess the maximum.



So right off the bat we guess that Shannon entropy is maximized when the probability is equally distributed to each letter in the alphabet. Lets see how we can prove this for a general alphabet. Witten prescribes using Lagrange multipliers with the constraint $\Sigma_i p_i=1$. So we want to solve the system of equations

$$\frac{dS}{dp_i}=-\frac{\ln p_i+1}{\ln 2}=\lambda=\frac{d(\text{Constraint})}{dp_i}\qquad\sum_{i=1}^{k}p_i=1$$

The blue term gives us

$$\ln p_i=\ln p_j\Rightarrow p_i=p_j\ \forall i,j\Rightarrow\sum_{i=1}^{k}p_i=kp=1\Rightarrow\boxed{p_i=\frac{1}{k}\ \forall i}$$

The Shannon Entropy is maximized if each letter has the same probability of occuring!

Now lets talk about two good friend, Dan and Phil. They're sending messages to each other. Dan sends an instance of a discrete random variable x to Phil who recieves an instance of a discrete random variable y. What all this means physically is that Dan is speaking a random letter into a really bad microphone and Phil is listening on really shitty headphones and can't tell with 100% certainty what Dan is saying. Let $p(x_i, y_j)$ be the probability that Dan says x_i and Phil hears y_j. Now lets say for a second that Phil is certain that he just heard y_j on his headphone. In this case the probability that Dan said x_i is given by Bayes rule

$$p(x_i|y_j) = \frac{p(x_i, y_j)}{p(y_j)} \Rightarrow S(x|y = y_j) = -\sum_i p(x_i|y_j)\log_2(p(x_i|y_j))$$

Knowing that Phil heard y_j for certain will make us more sure Phil can guess what Dan said and thus should lower the entropy. So we can guess that the Shannon entropy corresponding to the conditional probability distribution is the entropy of the message given that we heard y_j for certain. But in reality this will never happen since y is a random variable. But we can take a weighted sum of all these entropies to see on average what the entropy of the message Phil recieves is. To take this weighted sum we first need to reduce the dimension of the probability distribtuion to get $p(y)$

$$p(y) = \sum_i p(y, x_i)$$

And from here we can take a weighted sum

$$\begin{aligned}
\sum_j p(y_j)S(x|y = y_j) &= -\sum_j \sum_i p(y_j)p(x_i|y_j)\log_2(p(x_i|y_j)) \\
&= -\sum_i \sum_j p(y_j)\frac{p(x_i, y_j)}{p(y_j)}\log_2\left(\frac{p(x_i, y_j)}{p(y_j)}\right) \\
&= -\left(\sum_i \sum_j p(x_i, y_j)\log_2(p(x_i, y_j)) - \sum_i \sum_j p(x_i, y_j)\log_2(p(y_j))\right) \\
&= -\left(\underbrace{\sum_i \sum_j p(x_i, y_j)\log_2(p(x_i, y_j))}_{S_{XY}} - \underbrace{\sum_i p(y_j)\log_2(p(y_j))}_{S_Y}\right)
\end{aligned}$$

So we find that the average entropy of a message sent between Dan and Phil is the difference between the entropy of the joint probability distribution minus the entropy of the probability distribution of y. It turns out this little guy is so important that they have a name

> Definition 2: (Conditional Entropy) The entropy of some probability distribution x given that we have observed y is
>
> $$S_{XY} - S_Y$$

Conditional Entropy is related to another concept which is called **mutual information** denoted by $I(X;Y)$. This is the information that we can get about X given that we have observed Y. This would just be the difference of the actual entropy of x (S_X) and how much we don't know about x (S_{XY}).

Definition 3: (Mutual Information) Meaesurement of how much we know about a probability distribution x once we have observed y

$$I(X;Y) = S_X - S_{XY} + S_Y$$

Lets consider an example to motivate the definition of another cool guy.

Example 2: (Your Friend is a Sketchy Loser who likes Gambling and Dice) You and your friend are playing a game with two die. If the sum of the two die is > 7 you win a buck and if they're < 7 then your friend wins a buck. Rolling a 7 is a tie. We know that you both should have an equal chance of winning. However you're suspicious of your friend. He is a big chater after all. Even more suspicious, he insits on using his special die. He says he got them from a famous die maker on etsy for free after buying a space themed DnD dice set. First lets review what you'd expect to happen. There are 36 possible outcomes to this game.

$$\begin{aligned} \text{Your Friend Win}: &(1,1)(1,2)(1,3)(1,4)(1,5)(2,1)(2,2)(2,3)(2,4)(3,1)(3,2)(3,3)(4,1)(4,2)(5,1) \\ \text{You Win}: &(2,6)(3,5)(3,6)(4,4)(4,5)(4,6)(5,3)(5,4)(5,5)(5,6)(6,2)(6,3)(6,4)(6,5)(6,6) \end{aligned}$$

Let the outcome of the i^{th} game be denoted by x_i. If your friend isn't being a dick and cheating then we'd have the probability of $G(x_i)$ occuring as

$$G(x_i) = \begin{cases} \text{You Win} & \frac{15}{36} \\ \text{You Lose} & \frac{15}{36} \\ \text{Tie} & \frac{5}{36} \end{cases}$$

You and your friend play N games. For large N we'd expect you to have won $\frac{15}{36} \times N$, your friend to have won the same number of games, and to have tied $\frac{5}{36} \times N$ times. The number of sequences of N games with this many of each ending state is

$$\frac{N!}{\left(\frac{15}{36}N\right)!\left(\frac{15}{36}N\right)!\left(\frac{5}{36}N\right)!}$$

And thus we can judge the probability of what we've seen by considering [a]

$$\mathcal{P} = \underbrace{\left(\underbrace{\left(\frac{15}{36}\right)^{N\times\frac{15}{36}}}_{\text{You Win}}\underbrace{\left(\frac{15}{36}\right)^{N\times\frac{15}{36}}}_{\text{Your Friend Wins}}\underbrace{\left(\frac{5}{36}\right)^{N\times\frac{5}{36}}}_{\text{Tie}}\right)}_{\text{Probability you expect}}\underbrace{\frac{N!}{\left(\frac{15}{36}N\right)!\left(\frac{15}{36}N\right)!\left(\frac{5}{36}N\right)!}}_{\text{What you see}}$$

The first big parenthesis corresponds to the probability of the given sequence occuring. Each time x_i occurs we multiply by a factor of $G(x_i)$. Since for large N we expect x_i to occur $N \times G(x_i)$ times the resulting probability will be the first term. The second term is the number of sequences where each x_i occurs $N \times G(x_i)$ times. But what if the die are rigged (presumably in your friends favor unles he's just a real interesting guy)? What if instead your friend can manipulate the die so that the probability of each outcome $P(x_i)$ is

$$P(x_i) = \begin{cases} \text{You Win} & \frac{10}{36} \\ \text{You Lose} & \frac{22}{36} \\ \text{Tie} & \frac{3}{36} \end{cases}$$

You'd still expect the same probability based on $G(x_i)$ if you go into this thinking your friend is honest but after large N what you see will be different and will instead be caused by $P(x_i)$.

$$\mathcal{P} = \underbrace{\left(\underbrace{\left(\frac{15}{36}\right)^{N\times\frac{10}{36}}}_{\text{You Win}} \underbrace{\left(\frac{15}{36}\right)^{N\times\frac{22}{36}}}_{\text{Your Friend Wins}} \underbrace{\left(\frac{5}{36}\right)^{N\times\frac{3}{36}}}_{\text{Tie}}\right)}_{\text{Probability you expect}} \underbrace{\frac{N!}{\left(\frac{10}{36}N\right)!\left(\frac{22}{36}N\right)!\left(\frac{3}{36}N\right)!}}_{\text{What you see}}$$

How can we quantify this discrepency? That's where **relative entropy** comes in

[a] I'm still not too certain on what Witten means by "judge." I think it has something to do with trying to see the "distance" between two probability distributions but from what I can tell it seems like he just multiplied two probability distribution dependent quantities together and called it a day. TODO intuition.

Definition 4: (Relative Entropy) Let X be a random variable which denotes the outcome of an experiment. We have some theory that predicts the probability distribution for our outcome to be Q_X. However if our theory isn't quite on the mark and the actual probability distribution if P_X how could we guess that Q_X is wrong. Let x_i denote the final state of the i^{th} expierment. Also let there be s possible final states. We do N experiments where N is large. If we go in thinking that Q_X is correct then we will judge the probability of what we have seen with

$$\mathcal{P} = \underbrace{\left(\prod_{i=1}^{s} Q_X(x_i)^{P_X(x_i)\times N}\right)}_{\text{Probability you expect}} \times \underbrace{\frac{N!}{\prod_{j=1}^{s}(P_X(x_i)\times N)!}}_{\text{What you see}}$$

When defining shannon entropy we saw that the second term could be rewritten as

$$\underbrace{\frac{N!}{\prod_{j=1}^{s}(P_X(x_i)\times N)!}}_{\text{What you see}} \approx 2^{-N\sum_i P_X(x_i)\log_2(P_X(x_i))}$$

And the first term can be trivially rewritten as

$$\underbrace{\left(\prod_{i=1}^{s} Q_X(x_i)^{P_X(x_i)\times N}\right)}_{\text{Probability you expect}} = 2^{N\sum_i P_X(x_i)\log_2(Q_X(x_i))}$$

All together this gives us

$$\mathcal{P} \approx 2^{-N\sum_i P_X(x_i)\left(\log_2\left(\frac{P_X(x_i)}{Q_X(x_i)}\right)\right)} = 2^{-N S(P_X\|Q_X)}$$

Where the red term is what we define as **relative entropy** (or Kullback-Liebler divergence if you're in the mood for a mouthfull.)

$$S(P_X \parallel Q_X) = \sum_i P_X(x_i) \times \log_2\left(\frac{P_X(x_i)}{Q_X(x_i)}\right)$$

The relative entropy has a few properties. First we note that if $P_X = Q_X$ then relative entropy is zero, else it's positive. We'll also notice that if $\mathcal{P}$ decreases and N is fixed then our $S(P_X \parallel Q_X)$ is increasing. From this we can guess that larger relative entropy means the more sure our initial hypothesis Q_X is wrong. So what can relative entropy do for us?

Example 3: (What Relative Entropy can tell us about Mutual Information) Something that relative entropy can tell us is that mutual information is positive. Consider a joint probability distribution $P_{X,Y}(x,y)$. We can reduce the distribution to a single variable in the normal way

$$P_X(x) = \int P_{XY}(x,y)dy = \sum_j P_{XY}(x,y_j) \qquad P_Y(y) = \int P_{XY}(x,y)dx = \sum_j P_{XY}(x_j,y)$$

And we'll define a new probability distribution $Q_{XY}(x,y) = P_X(x)P_Y(y)$. So what have we done? What Q_{XY} is saying is that $P_X(x)$ and $P_Y(y)$ are statistically independent. But we don't know if this actually true or not. But recall that intuitivley, relative entropy is sort of like the distance between two probability distributions. So if we find that the relative entropy between Q_{XY} and P_{XY} is zero we can say that $P_X(x)$ and $P_Y(y)$ are statistically independent. So lets calculate the relative entropy

$$\begin{aligned}
&S(P_{XY} \parallel Q_{XY}) = \\
&= \sum_{i,j} P_{XY}(x_i,y_j) \times \log_2\left(\frac{P_{XY}(x_i,y_j)}{Q_{XY}(x_i,y_j)}\right) \\
&= \sum_{i,j} P_{XY}(x_i,y_j) \times \left(\log_2 P_{XY}(x_i,y_j) - \log_2 P_X(x_i) - \log_2 P_Y(y_j)\right) \\
&= \underbrace{\left(\sum_{i,j} P_{XY}(x_i,y_j)\log_2 P_{XY}(x_i,y_j)\right)}_{-S_{XY}} - \underbrace{\left(\underbrace{\sum_{i,j} P_{XY}(x_i,y_j)}_{\sum_i P_X(x_i)}\log_2 P_X(x_i)\right)}_{-S_X} - \underbrace{\left(\underbrace{\sum_{i,j} P_{XY}(x_i,y_j)}_{\sum_j P_Y(y_j)}\log_2 P_Y(y_j)\right)}_{S_Y} \\
&= S_X - S_{XY} + S_Y = I(x;y)
\end{aligned}$$

So we see that the mutual information is the relative entropy between a joint distribution P_{XY} and the hypothesis that P_X and P_Y are statistically independent and thus **must be positive.**

$$I(X;Y) = S_X + S_Y - S_{XY} \geq 0$$

Relative entropy has another interesting property called the **monoticity of relative entropy.**

Definition 5: (Monoticity of Relative Entropy) Lets say we have two joint probability distribution P_{XY} and Q_{XY} where P_{XY} is the "true" probability distribution and Q_{XY} is the hypothesized probability distribution. After N measurements we can quantify our certainty or uncertainty that our hypotheszied distribtuion Q_{XY} is correct with the relative entropy $S(P_{XY} \| Q_{XY})$. But lets say we're down bad. We can only measure X. So reducing to one variable gives us P_X and Q_X and again we can asses our hypothesis with the relative entropy $S(P_X \| Q_X)$. But lets think very hard for a second. Would $S(P_X \| Q_X)$ be greater than or less than $S(P_{XY} \| Q_{XY})$. Or physically we want to know, is our assesment of our hypothesis going to be more or less optimistic given that we've reduced the number of random variables[a].

$$S(P_{XY} \| Q_{XY}) \geq S(P_X \| Q_X)$$

We call this the monotonicity of relative entropy. To get a feel for why this is true. Consider observing a sequence of outcomes $\{x_{i_1}, \ldots, x_{i_n}\}$. For this sequence of outcomes there should also exists some $\{y_{i_1}, \ldots, y_{i_n}\}$ that minimizes the relative entropy. But any sequence of y we view will at best be equal to the relative entropy of only viewing one dimension of the joint probability distribution and will probably be increase the relative entropy. So the relative entropy of being able to measure one degree of freedom of the joint probability distribution must be lower than being able to measure the entire joint probability distribution.

[a] Witten just says "it is harder to disprove the inital hyptohesis if we observe only X" like it's clear but unfortunately I don't have any inspired thoughts that make this statement obvious intuitivley. My best guess is that reducing the number of random variables helps us cheat a bit sort of like looking at the first few lines of the solution to a problem. We've sort of assumed that we're good for one variable and are just trying to find the distance between two simpler distributions and thus $S(P_{XY} \| Q_{XY})$ is more chaotic then its single variable counterpart. Or maybe I'm overthinking it and there isn't an intuition for it. In other word's Witten is just quoting the result of the proof we're about to do without saying we should know this intuitively. Edit: nevermind he literally gives the intuition in the next paragraph.

If my ham-handed explanations and Witten's very nice intuition for the monotinicity of relative entropy don't satisfy you, don't fret! We'll also show it the brain dead way with some algebra. We can restate the monoticity of relative entropy as

$$S(P_{XY} \| Q_{XY}) - S(P_X \| Q_X) \geq 0$$

Which we can rewrite as

$$S(P_{XY} \parallel Q_{XY}) - S(P_X \parallel Q_X) = \sum_{i,j} P_{XY}(x_i,y_j) \times \left(\log_2\left[\frac{P_{XY}(x_i,y_j)}{Q_{XY}(x_i,y_j)}\right] - \log_2\left[\frac{P_X(x_i)}{Q_X(x_i)}\right]\right)$$

$$= \sum_{i,j} \underbrace{\frac{P_{XY}(x_i,y_j)}{P_X(x_i)} \times P_X(x_i)}_{\substack{P_X(x_i)P(y_j|x_i) \\ \text{by Bayes rule}}} \times \log_2\left[\underbrace{\frac{P_{XY}(x_i,y_j)}{P_X(x_i)}}_{\substack{P(y_j|x_i) \\ \text{by Bayes rule}}} \times \underbrace{\frac{Q_X(x_i)}{Q_{XY}(x_i,y_j)}}_{\substack{Q(y_j|x_i)^{-1} \\ \text{by Bayes rule}}}\right]$$

$$= \sum_i P_X(x_i) \underbrace{\sum_j P_X(y_j|x_i) \log_2\left[\frac{P(y_j|x_i)}{Q(y_j|x_i)}\right]}_{\substack{S(P_{Y|X=x_i} \| Q_{Y|X=x_i}) \\ \geq 0 \text{ since it's a} \\ \text{relative entropy}}}$$

$$= \sum_i P_X(x_i) S(P_{Y|X=x_i} \parallel Q_{Y|X=x_i}) \geq 0$$

There we go! Monoticity of relative entropy can show us something very interesting. Consider some probability distributions $P_{XYZ}(x_i, y_j, z_k)$ and $Q_{XYZ} = P_X(x_i)P_{YZ}(y_j, z_k)$ where we integrate out variables in the normal way. From the monoticity of relative entropy we know that

$$S(P_{XYZ} \parallel Q_{XYZ}) \geq S(P_{XY}|Q_{XY})$$

When we were proving that mutal information is positive we showed that

$$S(P_{XY} \parallel Q_{XY}) = S_X - S_{XY} + S_Y$$

We can apply this result here

$$\cancel{S_X} - S_{XYZ} + S_{YZ} \geq \cancel{S_X} - S_{XY} + S_Y$$

Giving us the result

$$S_{XY} + S_{YZ} \geq S_{XYZ} + S_Y$$

We call this strong subadditivity. From our definition of mutual information this is equivalent to saying that

$$I(X;YZ) \geq I(X;Y)$$

Which makes sense because being able to view the whole distribution will give you more information about X than only viewing a slice of the probability distribution.

Witten: Quantum Information Theory

Started: February 07, 2021. Finished: Probably Never :pensive:

Lets say we're studying a system A with Hilbert space $\mathcal{H}_A$ and let B be anything else relevant (or even the entire rest of the universe if you want to be spicy) with hilbert space $\mathcal{H}_B$. We can describe the joint Hilbert space with a tensor product $\mathcal{H}_{AB} = \mathcal{H}_A \otimes \mathcal{H}_B$ and vectors in this Hilbert space as $\psi_{AB} = \psi_A \otimes \psi_B$ where $\psi_I \in \mathcal{H}_I$. If ψ_{AB} is a unit vector then we can choose ψ_A and ψ_B to also be unit vectors. And thus from here we can ignore B completley when finding observables for the system we're studying, A. Consider some observable $\mathcal{O}_A$ which is an operator on $\mathcal{H}_A$. Then

$$\langle \psi_{AB} | \mathcal{O}_A \otimes 1_B | \psi_{AB} \rangle = \langle \psi_A | \mathcal{O}_A | \psi_A \rangle \langle \psi_B | \psi_B \rangle = \langle \psi_A | \mathcal{O}_A | \psi_A \rangle$$

However most days aren't good days. It's not usually the case where ψ_{AB} is the product state of two vectors ψ_A and ψ_B. Lets say $\dim(\mathcal{H}_A) = 2$ and $\dim(\mathcal{H}_B) = 3$. We then find that any pure state in $\mathcal{H}_{AB}$ can be written as a 2×3 matrix. Furthermore through some unitary transformations we can diagonalize the pure state so that it has the form

$$\psi_{AB} = \begin{bmatrix} p_1 & 0 & 0 \\ 0 & p_2 & 0 \end{bmatrix}$$

More generally we can write this as a Schmidt decomposition

Definition 6: (Schmidt decomposition) A pure state ψ_{AB} in the hilbert space $\mathcal{H}_A \otimes \mathcal{H}_B$ can be written in terms of a orthonormal basis ψ_A^i and ψ_B^i as

$$\psi_{AB} = \sum_i \sqrt{p_i} \psi_A^i \otimes \psi_B^i$$

Where $p_i > 0$ which we can interpret as probilities. We see that ψ_{AB} is a unit vector if $\sum_i p_i = 1$ (just consider $\psi_{AB}^* \psi_{AB}$).

Lets try using the Schmidt decomposition to find observables. Consider the same observable $\mathcal{O}_a \otimes 1_B$

$$\langle \psi_{AB} | \mathcal{O}_a \otimes 1_B | \psi_{AB} \rangle = \sum_i \sum_j \sqrt{p_i p_j} \langle \psi_A^i | \mathcal{O}_a | \psi_A^j \rangle \langle \psi_B^i | 1_B | \psi_B^j \rangle = \sum_i p_i \langle \psi_A^i | \mathcal{O}_a | \psi_A^i \rangle = \mathrm{Tr} \rho_A \mathcal{O}_A$$

Where in the last equality we defined the **density matrix**

Definition 7: (Density Matrix) The density matrix of a system A can be written as

$$\rho_A = \sum_i p_i |\psi_A^i\rangle \langle \psi_A^i|$$

We can see the matrix is hermitian. Also apprently it's obvious that this matrix is positive semi-definite. Furthermore we've asserted that $\sum_i p_i = 1$ (the probabilites have to sum to 1) meaning that $\mathrm{Tr}\rho_A = 1$.

Wilde Chapter 2: Classical Shannon Theory

Started: February 08, 2021. Finished: February 15, 2021

Lets start with an example in coding.

Example 4: (Efficient Coding) Suppose we have two friends, Dan and Phil. Dan wants to transmit messages to Phil through a noiseless bit channel. Dan also want to use this channel as little as possible since Dan is lazy. Now suppose Dan wants to transmit a message where each letter of the message is drawn from a probability distribution where

$$P(\{a,b,c,d\}) = \left\{\frac{1}{2},\frac{1}{8},\frac{1}{4},\frac{1}{8}\right\}$$

How efficiently can we encode a message if the probability distribution is the one we have above? The simplest way would be something like

$$\{a,b,c,d\} \xrightarrow{\text{encoding}} \{00,01,10,00\}$$

We can evaluate the efficiency of this coding by calculating the expected length of each letter transmitted. But if we do this then we realize that the encoding isn't as efficient as it could be. We're not exploiting the skewed probability distribution we have. Another encoding we could consider is

$$\{a,b,c,d\} \xrightarrow{\text{encoding}} \{0,110,10,111\}$$

With this encoding the message should still be readable (I guess you could make a state machine to confirm this) and we should have decreased the expected length of the message. This is because the expected value of each letter transmitted should have decreased. To see this we can look at

$$\frac{1}{2}\times 1+\frac{1}{8}\times 3+\frac{1}{8}\times 3+\frac{1}{4}\times 2=\frac{7}{4}<2$$

The example above leads us to consider the questions: how do we measure information? One way we could try to do it is consider at which symbol we would be most surprised to see. This leads us to the definition of information content

Definition 8: (Information Content) If the probability of some event happening is $p(x)$ then the information content of this event is

$$i(x)=\log_2\left(\frac{1}{p(x)}\right)=-\log_2(p(x))$$

Now consider the scenario where we measure two events x_1 and x_2 independently. Then the probability that we measure both these events is

$$p(x_1,x_2)=p(x_1)p(x_2)$$

Then the information content of these two events happening is

$$i(x_1,x_2)=-\log_2(p(x_1,x_2))=-\log_2(p(x_1)p(x_2))=-\log_2(p(x_1))-\log_2(p(x_2))=i(x_1)+i(x_2)$$

We call this property the *additivity* of information content. We can also calculate the expectation value of the information content of some information source. This quantity is important enough to have a name: **entropy**.

Definition 9: (Entropy) The entropy of an information source is the expected value of the information content of that source

$$\sum_x p(x)i(x) = -\sum_x p(x)\log_2(p(x))$$

We now have some tools to ask a very interesting question: is there a more efficient coding scheme than the one we came up with example 4? To consider this question we'll consider the general case. We can model a general information source as a random variable X with probability density $p_X(x)$ whose instances x we'll call *letters* of some alphabet χ. The entropy of this random variable is

$$H(X) = -\sum_{x\in\chi} p(x)\log(p_X(x))$$

And the information content of the random variable X is

$$i(X) = -\log(p_X(X))$$

Which might look strange. What we have defined is another random variable $i(X)$ that is a function of a random variable plugged into its own probability distribution. We'll see how this is useful in a bit. Now lets go into a quick aside

Example 5: (Entropy of a Uniform Random Variable) Lets consider a uniform random variable. This means that $p(x) = \frac{1}{|\chi|}$ for all $x \in \chi$. This means that

$$H(X) = -\sum_{x\in\chi} p(x)\log(p(x)) = -\frac{1}{|\chi|}\times|\chi|\times(-\log(|\chi|)) = \log|\chi|$$

Anyways back to the plot

One of Shannon's cool ideas for efficient coding was to let the information source emit a block of information, code this entire block (as opposed to letter by letter as we did in the example) with some possible error, and then show that this error vanishes for large block size. Lets try to make this concrete. Suppose the information source emits a sequence of letters

$$x^n = x_1 \dots x_n$$

We'll introduce the notation that X^n that denotes the random variable that is associated with the entire sequence x^n and X_i to denote the random variable associated with the i^{th} letter spit out by the information source. We'll assert that the information source is independent and identically distributed (IID) meaning that $X_i = X_j = X$ for all i, j and

$$p_{X^n}(x^n) = p_{X_1\dots X_n}(x_1,\dots,x_n) = p_{X_1}(x_1)\times\cdots\times p_{X_n}(x_n) = \prod_i p_X(x_i)$$

But we can go even further! First we'll introduce the notation that $N(a_i|x^n)$ is the number of times the letter $a_i \in \chi$ appears in the sequence x^n. For large n we can assume that

$$N(a_i|x^n) = n \times p_X(a_i)$$

And thus $p(x^n)$ can be simplified even more into

$$p(x^n) = \prod_{x \in \mathcal{X}} p_X(x)^{N(x|x^n)}$$

What this formula is also saying is that we can (if we squint our eyes a little) permute the block x^n into a nicer sequence

$$x^n \to \underbrace{a_1 \dots a_1}_{N(a_1|x^n)} \dots \underbrace{a_{|\mathcal{X}|} \dots a_{|\mathcal{X}|}}_{N(a_{|\mathcal{X}|}|x^n)}$$

So far what we've done is derive properties of a instance of X^n. From here we'll try to use the result we have so far to derive some nice result about the more general X^n. The first thing we'll consider is the (currently mysteriously named) *sample entropy*

Definition 10: (Sample Entropy) For a random variable X^n that corresponds to a IID information source that generates a sequence $x^n = x_1 \dots x_n$ we define the sample entropy as the average information content of each letter in the random sequence. Namely

$$\frac{1}{n} \times i(X^n) = -\frac{1}{n}\log(P_{X^n}(X^n))$$

We can use the natural generalization of the result above to derive a more suggestive form of sample entropy

$$\begin{aligned} -\frac{1}{n}\log(P_{X^n}(X^n)) &= -\frac{1}{n}\log\left(\prod_{x\in\mathcal{X}} p_X(x)^{N(x|X^n)}\right) \\ &= -\frac{1}{n}\sum_{x\in\mathcal{X}}\log\left(p_X(x)^{N(x|X^n)}\right) \\ &= -\sum_{x\in\mathcal{X}}\frac{N(x|X^n)}{n}\log(p_X(x)) \end{aligned}$$

Now notice that for large n we can approximate $N(x|X^n) = n \times p_X(x)$. Which means that we can write for large n that

$$-\frac{1}{n}\log(P_{X^n}(X^n)) \approx -\sum_{x\in\mathcal{X}} p_X(x)\log(p_X(x)) = H(X)$$

But what's this? Isn't the RHS just the entropy of the random variable X? So what we have shown is that for large n, the information source emits a sequence whose sample entropy is close to true entropy. Wilde started saying some weird things about empircal distributions and starting whipping out our favorite $\epsilon - \delta$ arguments but I think the above is the core of what he's saying. Just for future reference the punchline of his argument was that

$$\lim_{n\to\infty} \Pr\{|(\text{Sample Entropy}) - (\text{True entropy of } X)| \leq \delta\} = 1 \quad \forall \delta > 0$$

Now this leads us to a definition

Definition 11: (Typical Sequence, Typical Set, and their properties) We define a typical sequence as a instance of X^n which we'll denote as x^n whose sample entropy is near the true entropy (it's not given in the book but I assume for some $\delta > 0$ we want the difference to be less than δ.) The typical set is the set of all typical sequences. For now we'll assert without proof that the size of the typical set is $\approx 2^{nH(X)}$. The typical set has a few properties

(a) The probability that a emitted sequence by the information source is in the typical set approaches 1 for large n

(b) The size of a typical set is exponentially smaller the the size of the set containing all possible sequences.

(c) The probablity of a particular typical sequence is roughly uniform and is $\approx 2^{-nH(X)}$.

(a), (b), and (c) make up what is called the asymptotic equipartition theorem.

Lets return to our original motivating problem, trying to come up with efficient coding schemes. Given the notion of a typical set we could try a coding scheme that's a bijective function from the typical set whose cardinality is $2^{nH(s)}$ to some binary string of length $nH(X)$. If we encounter a sequence that is not part of the typical set (the probability of this is vanishingly small for large n) then we throw an error! Lets try to evaluate the efficiency of this coding scheme. We can do this with a *compression rate*

Definition 12: (Compression Rate) The compression rate is defined as

$$(\text{compression rate}) = \frac{(\text{\# of noiseless channel bits needed})}{(\text{\# of source symbols})}$$

The compression rate of this typical set block coding scheme is

$$\frac{nH(X)}{n} = H(X)$$

This also gives us a operational definition of the shannon entropy as the compression ratio of our coding scheme. We won't prove this yet but this is as good as we can do.

Proving a coding theorem is usually broken down into two parts. First the direct coding theorem where we construct a coding scheme with some specific compression rate. This is meant to show that

$$\begin{array}{c}\text{Rate of compression greater}\\ \text{than entropy of source}\end{array} \rightarrow \begin{array}{c}\text{Exists a coding scheme}\\ \text{that can achieve losless compression}\end{array}$$

Where here loselss compression means probablilty of error in decoding is small. Then we have to prove the converse

$$\begin{array}{c}\text{Exists a coding scheme}\\ \text{that can achieve losless compression}\end{array} \rightarrow \begin{array}{c}\text{Rate of compression greater}\\ \text{than entropy of source}\end{array}$$

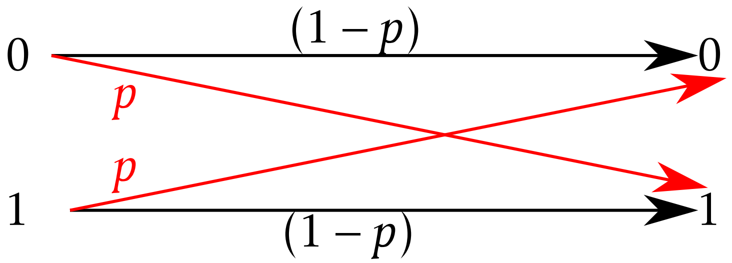


Figure 1: Illustration of a classical noisy bit-flip channel

We've had a lot of fun in noiseless channels so far so lets finally start talking about noise. In particular we'll talk about a bit-flip channel. When we send a bit, the probability that the bit is flipped is $(1-p)$. This is illustrated in Figure 1. Also assume that the output of this channel is IID. Again lets consider our two favorite people who send messages to eachother, Dan and Phil. The found this noisy transmitter that's cheaper than the noiseless one (in reality there probably aren't any noiseless ones) and they want to figure out how they can efficiently and without error send messages through this channel. If they just use the channel as is then there is a probability p that Phil misinterprets what Dan is saying. What about if we try a coding scheme like

$$0 \to 000 \quad 1 \to 111$$

And then take a majority vote out of the triplets to decode what the triplet actually is. For example 110 would decode to 1 and 001 would decode to 0. Lets analyze this coding scheme. First if we want to trasmit a 0 then we have

$$\begin{aligned}
\text{Output} &: \text{Probability}\\
\{(000)\} &: (1-p)^3\\
\{(001),(010),(100)\} &: 3\times(1-p)^2\times p\\
\{(110),(101),(011)\} &: 3\times(1-p)\times p^2\\
\{(111)\} &: p^3
\end{aligned}$$

And we have a similar table for wanting to transmit a 1. So what's the probability of error here? Well it would just be

$$p(e) = p(0)p(e|0) + p(1)p(e|1)$$

From the above table we can read off

$$p(e|0) = 3(1-p)p^2 + p^3 = 3p^2 - 2p^3 \xrightarrow{by symmetry} p(e|1) = 3p^2 - 2p^3 = p(e|\cdot) = p(e|0)$$

Thus giving us

$$p(e) = p(e|\cdot)\left(\underbrace{p(0)+p(1)}_{=1}\right) = 3p^2 - 2p^3$$

So now the most natural question to ask is, when is this naive coding scheme better than just not doing anything at all? We can formalize this as

$$p(e) < p \Rightarrow 3p - 2p^2 < 1 \to 0 < 2p^2 - 3p + 1$$

Solving this for p yields

$$p > \frac{3 \pm \sqrt{9-8}}{4} = \frac{3 \pm 1}{4}$$

We can throw out the $p > 1$ since that's unphysical leaving us with $p > 1/2$. So if the probability flipping a bit is more than $\frac{1}{2}$ the above coding scheme is better than doing nothing. However if $p < \frac{1}{2}$ the coding scheme is worse than doing nothing at all[1]. In general the rate of error decreases as we increase the redundancy of our code but do you really want to do it that way? Lets look for a more aesthetically pleasing coding scheme. To do this lets try solving a more general problems(that's a good trick.) We'll first need a more general model of a noisy channel, we'll call it the *discrete memoryless channel*

Definition 13: (Discrete Memoryless Channel) Dan wants to transmit to Phil a message $m \in M$ where M is a finite set of possible messages that Dan samples with uniform probability. To generalize the "bit-flip noise" we saw before we can introduce a conditional probability distribution $p_{Y|X}(y|x)$ for a random variable X and Y where we'll input a instance of X and the channel will poop out an instance of Y. We'll denote a sequence of messages as $x^n = x_1 \dots x_n$ and the random variable associated with x^n and x_i to be X^n and X_i with similarl notation for y_n and Y^n. We'll assume the channel is IID. The conditional probability distribution then becomes

$$p_{Y^n|X^n} = p_{Y_1|X_1}(y_1|x_1) \times \cdots \times p_{Y_n|X_n}(y_n|x_n) = \prod_{i=1}^{n} p_{Y|X}(y_i|x_i)$$

Any message $m \in \mathcal{M}$ Dan wants to send can be encoded in $b = \log_2(|\mathcal{M}|)$ bits and the actual instantiationi of this message in bits (which may or may not be the same as b) we'll denote as $x^n(m)$. The decoding of the recieved string of bits y^n by Phil we'll put off for now. The rate of this channel is

$$(\text{rate}) = \frac{\text{(numbers of bits needed to encode messag)}}{\text{(number of bits used to transmit message over channel)}}$$

So in our case we have

$$(\text{rate}) = \frac{1}{n} \times \log_2(|\mathcal{M}|)$$

How do we evaluate the performance of some coding scheme? Lets define $\mathcal{C} = \{x^n(m) | m \in \mathcal{M}\}$ and $p_e(m, \mathcal{C})$ be the probabilty that something goes wrong when using the coding scheme $\mathcal{C}$ when we transmit message m. The average and max error then is

$$\overline{p_e}(\mathcal{C}) = \frac{1}{|\mathcal{M}|} \sum p_e(m, \mathcal{C}) \qquad p_e^*(\mathcal{C}) = \max_{m \in \mathcal{M}} p_e(m, \mathcal{C})$$

The first thing that should be obvious is the fact that if the max probability of error is small then the average probability of error is small. What might be less obvious is that if the average probability of error is small then max probability is small for at least half the messages. Lets

[1] just like boarding groups for flights >:(

show this. We want to prove

$$\frac{1}{|M|}\sum_m p_e(m,\mathcal{C}) \leq \epsilon \to p_e(m,\mathcal{C}) \leq 2\epsilon \text{ for at least half the messages } m$$

Suppose not. Then there exists $N < |M|/2$ messages such that $p_e(m,\mathcal{C}) \leq 2\epsilon$. Lets set the N messages with $p_e \leq 2\epsilon$ to be zero and the $|M|-N$ messages with $p_e > 2\epsilon$ equal to two epsilon. Namely

$$\frac{1}{|M|}\sum_m p_e(m,\mathcal{C}) > \frac{|M|-N}{|M|} \times 2\epsilon = 2\epsilon - \frac{N}{|M|} \times 2\epsilon$$

Now note that $N < |M|/2$ meaning that $-N/|M| > -1/2$ giving us

$$\frac{1}{|M|}\sum_m p_e(m,\mathcal{C}) > \epsilon \text{ which is not } \leq \epsilon$$

Thus proving the statement by contradiction[2]

So up until now our general channel was two sepearte layers of randomness. The first layer of randomness is the choosing of the channel. The second layer of randomness is the noise in the channel when we send messages. It turns out that to prove that there exists a reliable coding scheme that is maximally efficient it's easier to introduce a third layer of randomness! This randomness will be in the actual instances of X that make up a codeword of m. Namely we'll have

$$P(X^n(m) = x^n(m)) = \prod_{i=1}^{n} P_X(x_i(m))$$

What's essentially what's happened is that the coding itself has become a random varaiable. $\mathcal{C}$ is no longer a map from $m \in M$ to some x^n but instead a probability distribution that each $m \in M$ will map to some specific x^n. A speicific instantiation of $\mathcal{C}$ which we'll now call $\mathcal{C}_0$ is what we had before, a map from $m \in M$ to some codeword. Furthermore each message will have the exact same probability distribution of x^n since there is no explicit reference to the message m, there is only a reference to what instance of X is in the slot x_i. So we now can write the probability that some coding scheme $\mathcal{C}_0$ arises as

$$P(\mathcal{C}_0 = \{x^n(m) | m \in \mathcal{M}\}) = \prod_{m}^{|\mathcal{M}|}\prod_{i=1}^{n} P_X(x_i(m))$$

So why is all this nonsense useful? Well what we can now consider is the expectation value of the mean probability of error over our space of random coding schemes. Namely

$$\mathbb{E}\left\{\frac{1}{|M|}\sum_{i=1}^{|M|} p_e(m,\mathcal{C})\right\}$$

[2] Okay I feel a bit skteched out by this proof of mine. But it did get around the need to use Markov's inequality like Wilde wanted us to. I'll leave it here for now.

We can use the linearity of expectation here to then write the above quantity as equal to

$$= \frac{1}{|M|}\sum_{i=1}^{|M|}\mathbb{E}\{p_e(m,\mathcal{C})\}$$

Now notice that our probability of error over all our possible coding schemes is independent of the actual message since the probability distribution of x^n for each m is the same(see above).. Thus we can replace m with any specific message we could transmit. This means that

$$\mathbb{E}\left\{\frac{1}{|M|}\sum_{i=1}^{|M|}p_e(m,\mathcal{C})\right\} = \mathbb{E}\{p_e(\mathcal{C})\}$$

Shannon then gives us a proof that we'll go into later[3] that says for some $\epsilon > 0$ which shrinks as block size increases we have

$$\mathbb{E}\{\overline{p}_e(\mathcal{C})\} \leq \epsilon$$

And thus by nature of expectation value we then know that there must exsist some coding scheme $\mathcal{C}_*$ such that

$$\overline{p}_e(\mathcal{C}_*) \leq \epsilon$$

We also proved by contradicition earlier that

$$\frac{1}{|M|}\sum_{m}p_e(m,\mathcal{C}) \leq \epsilon \to p_e(m,\mathcal{C}) \leq 2\epsilon \text{ for at least half the messages } m$$

We can now apply this result here to this special $\mathcal{C}_*$ to obtain a bound on the error of each message. First what we do is throw out all the message with $p_e(m,\mathcal{C}_*) > 2\epsilon$ which is half the messages[4]. For large n this has no effect on the rate. Recall that the number of messages $m \in \mathcal{M}$ we can send is 2^{nR}. Throwing out half the possible messages in $\mathcal{M}$ will lead to $2^{nR-1} = 2^{n(R-1/n)}$ where the red stuff is our new rate. As $n \to \infty$ we can see that the red stuff goes towards R. With this we can now put a bound on the maximum error of the coding (for our thanos snapped set of messages)

$$p_e(\mathcal{C}_0) \leq 2\epsilon$$

Now it's time to wave our hands. This will help us get to where we want to go. For the noiseless case we had the idea of a typical set. In the noisy channel case we'll also define something similar. Suppose Dan picks some sequence x^n that is a typical sequence. The probability that Phil recieves some sequence y^n is govenerned by the probability distribution $p_{Y|X}(y|x)$. Wilde also introduces (in the vaguest way possible) conditional entropy here. However in this case we're only told that it's denoted as $H(Y|X)$ and is a measure of how uncertain we are of a random variable Y given that we know a random variable X. Now with all these tools we can guess our way into a situation that for a given input sequence x^n there exists a typical set of output sequences governed by the conditional probability distribution as well as a corresponding conditional asymptotic equipartition theorem. We also know that the "conditional" typical set has the properties

[3] Why is Wilde such a tease

[4] I need to think more about this but why can we do this? Is it because we're picking messages uniformly meaning that any specific message we want to send we can send by making the message we want to send be at least half of the possibilities we can choose from? Or maybe it's because we're picking uniformly that makes the message an "arbitrary message." I dunno.

(a) The typical set of output sequences y^n has almost all the probabilities

(b) It's size is $\approx 2^{nH(Y|X)}$

(c) The probability of any given typical sequence given that we've sent x^n is $2^{-nH(Y|X)}$

Basically what we have now is a typical set but with conditional probabilities. If we so wish we can integrate out the the conditional probability to come up with a input independent typical set of outputs by doing

$$p_Y(y) = \sum_x P_{Y|X}(y|x) p_X(x) \Rightarrow \text{ have } H(Y) \Rightarrow \text{size of output typical set } = 2^{nH(Y)}$$

Ok so now lets actually imagine we're transmitting a code. Dan sends some x^n to Phil who recieves some y^n. Then Phil first checks if his recieved y^n is a typical sequence. If it isn't he throws an error and if not he then tries to determine which conditional typical set this y^n belongs to. If he guesses wrong then there's an error. This implies that we want as little overlap as possible between different conditional typical sets. In otherewords it might be good to divide up the input independent typical output set into as many disjoint conditional typical output sets as possible. In more words this means that we want

$$2^{M=nR} \approx \frac{2^{nH(Y)}}{2^{nH(Y|X)}} = 2^{n(H(Y)-H(Y|S))}$$

This allows us to bound the rate R with what we'll define as the mutual information

Definition 14: (Mutual Information)

$$I(X;Y) = H(Y) - H(Y|X) = \text{Mutual Information}$$

So now we have the bound that to minimize the decoding error by Phil we'd want the bound

$$R < I(X;Y) = H(Y) - H(Y-X)$$

Wilde again says that we'll prove later that that information content is concave wrt $P_X(x)$ meaning that there exists some P_X^* that maximizes information content. This leads us to the definition of the capacity of a channel

Definition 15: (Capacity) The capacity of the channel is the maximum rate of a channel. We can write the capacity of a channel $\mathcal{N}$ as

$$C(\mathcal{N}) = \max_{p_X(x)} I(X;Y)$$

Note that so far we've only proven that

$$\begin{array}{c}\text{If the rate of communication is}\\ \text{less than the channel capacity}\end{array} \Rightarrow \begin{array}{c}\text{Exists a reliable coding where}\\ \text{error vansihes as } n \to \infty\end{array}$$

In the future we'll turn the ⇒ into a ⇔[5]

[5] the last page of notes for this chapter I wrote in a very hazy mood so things feel a bit loose

Wilde Chapter 3: Noiseless Quantum Theory

Started: February 17, 2021. Finished:

We'll start in this chapter by talking about qubits which physically are two-level quantum systems. From there we'll move to the natural generalization which we'll call qudits which are $d-$level quantum systems. In this chapter we'll focus on noiseless quantum systems which are closed and idealized systems and study these systems by introducing the postulates of quantum mechanics.

We'll start by denoting the two levels of our quantum system as $|0\rangle$ and $|1\rangle$. Any superposition of these two states is a valid state

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \text{ where } |\alpha|^2 + |\beta|^2 = 1$$

These states lie on the unit sphere or a bloch sphere.

gonna skim this part since it's mostly introducing dirac notation

Besides the most naive choice for an orthonormal basis, we could also choose the useful basis (which we'll call the diagonal basis)

$$|+\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} \text{ and } |-\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}$$

most of this is covered in more detail in Littlejohn's notes so if you wanna reference look at his notes

Lets consider the NOT gate X which switches the bases.

$$X(\alpha|0\rangle + \beta|1\rangle) = \alpha|1\rangle + \beta|0\rangle$$

We can calculate the matrix form of this operator by evalauting matrix elements explicitly

$$\langle 0|X|0\rangle = 0 \quad \langle 0|X|1\rangle = 1 \quad \langle 1|X|0\rangle = 1 \quad \langle 1|X|1\rangle = 0 \Rightarrow X = \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$$

Lets also consider what happens for the diagonal basis

$$X|+\rangle = (+1)|+\rangle \quad X|-\rangle = (-1)|-\rangle \Rightarrow X = \begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$$

If we wanted to we could also define an operator Z where

$$Z|+\rangle = |-\rangle \quad Z|-\rangle = |+\rangle$$

If we choose the $|0\rangle, |1\rangle$ basis then in addition to the X and Z opetors we can define the I opertor (identity) nad $Y = iXZ$ operator (bit and phase flip.) The set of $\{I, X, Y, Z\}$ are the Pauli matrices. Just on examination we can see they're hermitian, unitary, square to identity, and have eigenvalues ±1 (write some code to grind throug h the algebra if you don't believe me.) Something more interesting we could considier are the eigenstates of Y. We could solve for them like we're mathematicaless high school students. But instead we'll just use mathematica. First note that

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

The eigenkets of Y in the computational basis are

$$|y_1\rangle = \begin{bmatrix} i \\ 1 \end{bmatrix} \text{ and } |y_2\rangle = \begin{bmatrix} -i \\ 1 \end{bmatrix}$$

The Pauli matrices have our favorite structure constant the levi-civita tensor.

We can also consider the Hadamard Gate which maps $|0\rangle \rightarrow |+\rangle$ and $|1\rangle \rightarrow |-\rangle$. We can read off the form of this operator H as

$$H = \{|+\rangle\langle 0|\} + \{|-\rangle\langle 1|\}$$

We can find the matrix elements in the computational basis of this operator decently quickly

$$\langle 0|H|0\rangle = \langle 0|+\rangle = \frac{1}{\sqrt{2}} \quad \langle 0|H|1\rangle = \langle 0|-\rangle = \frac{1}{\sqrt{2}} \quad \langle 1|H|0\rangle = \langle 1|+\rangle = \frac{1}{\sqrt{2}} \quad \langle 1|H|1\rangle = \langle 1|-\rangle = -\frac{1}{\sqrt{2}}$$

From the matirx representation we see that $HH = I$. Lets try to confirm this with the operator definition[6]. Using the orthognoal basis we can pretty quickly see that $HXH = Z$. Furthermore since H is it's own inverse the property $HZH = X$ follows immediately.
The last thing we'll talk about is the rotation operator.

$$R_{(X,Y,Z)}(\phi) = \exp\{i(X,Y,Z)\phi/2\}$$

Suppose a hermitian operator can be written as

$$A = \sum_i a_i |i\rangle\langle i|$$

The function of a operator then can be written as

$$f(A) = \sum_i f(a_i)|i\rangle\langle i|$$

Let $A \in \{X, Y, Z\}$. We can show that

$$R_A(\phi) = \exp\{iA\phi/2\}$$

Proof.

$$R_A(\phi) = 1 + iA\phi/2 - \frac{1}{2!}A^2\phi^2/4 - \frac{i}{3!}A^3\phi^2/8 + \ldots$$

We proved (when working through Littlejohn's quantum mechanics course) that

$$\sigma_i\sigma_j = \delta_{ij} + i\epsilon_{ijk}\sigma_k \Rightarrow \sigma_i\sigma_i = A^2 = 1$$

[6] todo

This gives us

$$R_A(\phi) = 1 + iA\phi/2 - \frac{1}{2!}\phi^2/4 - \frac{i}{3!}A\phi^3/8 + \dots$$

Now we can read off the series expansion of cos and sin

$$R_A(\phi) = \cos\left(\frac{\phi}{2}\right) + iA\sin\left(\frac{\phi}{2}\right) \qquad \square$$

A key part of quantum mechanics is the uncertainty principle. For two operators Z and X we have

$$\Delta Z \Delta X \geq \frac{1}{2}\left|\langle\psi|[X,Z]|\psi\rangle\right|$$

Notice that the lower bound depends on the state $|\psi\rangle$. Is there a state that makes the lower bound vanish? On examination we see that if $|\psi\rangle$ is a simultaneous eigen-ket of X and Z then the lower bound vanishes. However X and Z and have simultaneous eigen-kets. But recall the form of the qubit written in terms of the bloch sphere angles

$$|\psi\rangle = \cos(\theta/2)\begin{bmatrix}1\\0\end{bmatrix} + \sin(\theta/2)e^{i\phi}\begin{bmatrix}0\\1\end{bmatrix}$$

After a back-breaking 3 lines of mathematica we find that

$$\langle\psi|[X,Z]|\psi\rangle = -4i\cos(\theta/2)\sin(\theta/2)\sin(\phi)$$

If $\theta = 0, \pi$ or if $\phi = 0$ then we have a vanishing lower bound. On the bloch sphere this means as long as we have real coefficients or we're at one of the poles then the lower bound vanishes.

Definition 16: (Tensor Product) For concrete vectors the tensor product can be defined as follows

$$\begin{bmatrix}a_1\\b_1\end{bmatrix} \otimes \begin{bmatrix}a_2\\b_2\end{bmatrix} = \begin{bmatrix}a_1\begin{bmatrix}a_2\\b_2\end{bmatrix}\\ b_1\begin{bmatrix}a_2\\b_2\end{bmatrix}\end{bmatrix} = \begin{bmatrix}a_1a_2\\a_1b_2\\b_1a_2\\b_1b_2\end{bmatrix}$$

And similarly for a matrix

$$A \otimes B = \begin{bmatrix}a_{11} & a_{12}\\a_{21} & a_{22}\end{bmatrix} \otimes \begin{bmatrix}b_{11} & b_{12}\\b_{21} & b_{22}\end{bmatrix} = \begin{bmatrix}a_{11}B & a_{12}B\\a_{21}B & a_{22}B\end{bmatrix} = \text{you get the idea}$$

From the above definition we can represent a two qubit system as a tensor product of individual qubits. This extends to the basis of our Hilbert space of two qubit systems

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{bmatrix}1\\0\\0\\0\end{bmatrix} \qquad |01\rangle = |0\rangle \otimes |1\rangle = \begin{bmatrix}0\\1\\0\\0\end{bmatrix} \text{ etc...}$$

Similarly for multiple qubit systems we can tensor product operators together. Say we want to flip the first qubit and keep the second qubit the same. This operator could be written as

$$X \otimes I = \begin{bmatrix}0 & 0 & 1 & 0\\0 & 0 & 0 & 1\\1 & 0 & 0 & 0\\0 & 1 & 0 & 0\end{bmatrix}$$

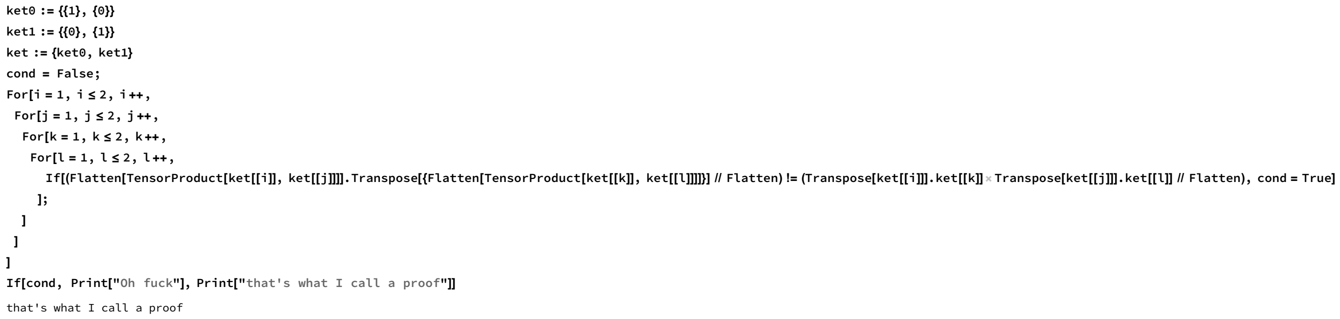
Honestly I feel like introducing abstract index notation here would make this section of the book less clunky. Another important property when calculating probability amplitudes for composite systems is

$$\langle \phi_1 \psi_1 | \phi_0 \psi_0 \rangle = \langle \phi_1 | \phi_0 \rangle \langle \psi_1 | \psi_0 \rangle$$

Where ϕ_i and ψ_j are arbitrary states. By linearity the above can be proven by showing

$$\langle ij | kl \rangle = \langle i | k \rangle \langle j | l \rangle \text{ for } i, j, k, l \in \{0, 1\}$$

This should be intuitivley clear. If not I'm sure there's some fancy proof for this. However clever proofs are for nerds and we can brute force this with mathematica.



```
ket0 := {{1}, {0}}
ket1 := {{0}, {1}}
ket := {ket0, ket1}
cond = False;
For[i = 1, i ≤ 2, i ++,
 For[j = 1, j ≤ 2, j ++,
  For[k = 1, k ≤ 2, k ++,
   For[l = 1, l ≤ 2, l ++,
     If[(Flatten[TensorProduct[ket[[i]], ket[[j]]]].Transpose[{Flatten[TensorProduct[ket[[k]], ket[[l]]]]}] // Flatten) != (Transpose[ket[[i]]].ket[[k]] × Transpose[ket[[j]]].ket[[l]] // Flatten), cond = True]
    ];
  ]
 ]
]
If[cond, Print["Oh fuck"], Print["that's what I call a proof"]]
that's what I call a proof
```

Running the above code confirms the necessary condition.

Lets talk about a important two bit operator, the CNOT gate

Definition 17: (CNOT Gate) In a classical setting the CNOT gate flips the second bit conditional on whether the first bit is set or not

$$00 \longrightarrow 00 \quad 01 \longrightarrow 01 \quad 10 \longrightarrow 11 \quad 11 \longrightarrow 10$$

In this case the gate is *conditional* on whether the first bit is set or not. In the quantum case we can represent the CNOT gate as

$$\text{CNOT} \equiv |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X$$

And in this case the second operation is *controlled* on the basis state of the first qubit. This gives us the name Controlled-NOT.

This can be generalized for any single qubit unitary U

Definition 18: (Controlled-U Gate) This is the generalization for the CNOT gate for a uniatry operator U which applies U to the second qubit if the first qubit is 1. We can represent this operator as

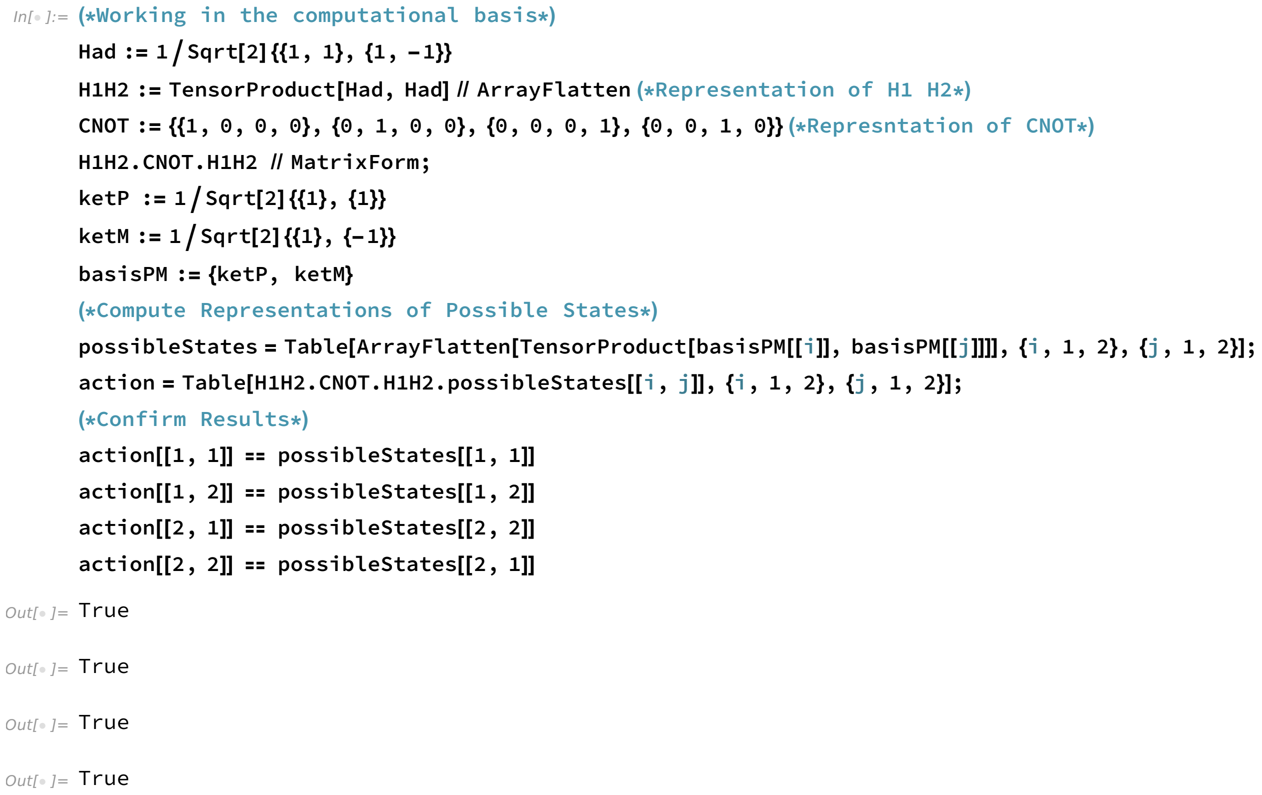
$$\text{controlled-}U \equiv |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U$$

And in general for an orthonormal bases $\{|\phi\rangle, |\psi\rangle\}$we can define

$$|\psi\rangle\langle\psi| \otimes I + |\phi\rangle\langle\phi| \otimes U$$

Lets see what kind of fun we can have with these guys.

Example 6: (Relation between Hadamards and CNOT) Something interesting to note is that if we apply Hadamards to the first and second qubit before and after applying a CNOT then we get a CNOT gate in the + and − basis. Lets show this explicitly. Working in the computational basis we know the representation of H and $|+\rangle$ and $|-\rangle$. So we can just plug things into mathematica to confirm things



```
In[ ]:= (*Working in the computational basis*)
        Had := 1/Sqrt[2]{{1, 1}, {1, -1}}
        H1H2 := TensorProduct[Had, Had] // ArrayFlatten (*Representation of H1 H2*)
        CNOT := {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 0, 1}, {0, 0, 1, 0}} (*Represntation of CNOT*)
        H1H2.CNOT.H1H2 // MatrixForm;
        ketP := 1/Sqrt[2]{{1}, {1}}
        ketM := 1/Sqrt[2]{{1}, {-1}}
        basisPM := {ketP, ketM}
        (*Compute Representations of Possible States*)
        possibleStates = Table[ArrayFlatten[TensorProduct[basisPM[[i]], basisPM[[j]]]], {i, 1, 2}, {j, 1, 2}];
        action = Table[H1H2.CNOT.H1H2.possibleStates[[i, j]], {i, 1, 2}, {j, 1, 2}];
        (*Confirm Results*)
        action[[1, 1]] == possibleStates[[1, 1]]
        action[[1, 2]] == possibleStates[[1, 2]]
        action[[2, 1]] == possibleStates[[2, 2]]
        action[[2, 2]] == possibleStates[[2, 1]]

Out[ ]= True

Out[ ]= True

Out[ ]= True

Out[ ]= True
```

Now lets talk about the no-cloning theorem

Definition 19: (No-Cloning Theorem) The No-Cloning theorem says that we cannot create a *universal cloner* that can creates a clone of any state.

Here's a simple proof by contradiction for this. Suppose there exists an arbitrary two qubit unitary opereator U which copies the first qubit into the second qubit(e.g. it's a universal cloner.) Let $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ be an arbitrary qubit state. The action of U would then be

$$\begin{aligned} U|\psi\rangle|0\rangle &= |\psi\rangle|\psi\rangle \\ &= (\alpha|0\rangle + \beta|1\rangle)(\alpha|0\rangle + \beta|1\rangle) \\ &= \alpha^2(|0\rangle)^2 + \alpha\beta|0\rangle|1\rangle + \beta\alpha|1\rangle|0\rangle + \beta^2(|1\rangle)^2 \end{aligned}$$

However U must also be linear and thus by linearity

$$\begin{aligned} U|\psi\rangle|0\rangle &= U(\alpha|0\rangle + \beta|1\rangle)|0\rangle \\ &= \alpha|0\rangle|0\rangle + \beta|1\rangle|1\rangle \end{aligned}$$

But this result and the previous result are in contradiction with eachother since there exists an α and β where the two do not equal. Thus there can not exist a universal cloner.

This theorem has applications in quantum cryptography since it guarentees attackers cannot clone the state used to establish secret keys.

Something that we should notice is that the above theorem doesn't imply that there can't be that can copy specific quantum states. Lets consider the following state

Example 7: (Cloner for Orthogonal States) Suppose we have two orthogonal states $|\psi\rangle$ and $|\phi\rangle$ where $\langle\psi|\phi\rangle = 0$. We can explicitly construct a operator U such that clones these two states

$$U|\psi\rangle|0\rangle = |\psi\rangle|\psi\rangle \quad U|\phi\rangle|0\rangle = |\phi\rangle|\phi\rangle$$

To do this lets go to a orthonormal basis $\{|a\rangle, |b\rangle\}$ where one of the basis vectors $\{|a\rangle\}$ is aligned along $|\psi\rangle$. Since we're considering a qubit state with dimensionality two we can write

$$|\psi\rangle = \psi|a\rangle = \begin{bmatrix}\psi\\0\end{bmatrix} \quad |\phi\rangle = \phi|b\rangle = \begin{bmatrix}0\\\phi\end{bmatrix} \quad |0\rangle = c|a\rangle + d|b\rangle = \begin{bmatrix}c\\d\end{bmatrix}$$

In this matrix form we see that we want

$$|\psi\rangle|0\rangle = \begin{bmatrix}c\psi\\d\psi\\0\\0\end{bmatrix} \xrightarrow{U} |\psi\rangle|\psi\rangle = \begin{bmatrix}\psi^2\\0\\0\\0\end{bmatrix}$$

$$|\phi\rangle|0\rangle = \begin{bmatrix}0\\0\\c\phi\\d\phi\end{bmatrix} \xrightarrow{U} |\phi\rangle|\phi\rangle = \begin{bmatrix}0\\0\\0\\\phi^2\end{bmatrix}$$

By looking at the form of these two we can see in this $\{|a\rangle, |b\rangle\}$ basis a unitary operator that could work is

$$U = \begin{bmatrix}\psi/c & 0 & 0 & 0\\0 & 0 & 0 & 0\\0 & 0 & 0 & 0\\0 & 0 & 0 & \phi/d\end{bmatrix}$$

Related to the No-Cloning theorem is something called the No-Deletion theorem

Definition 20: (No-Deletion Theorem) We have two copies of a quantum state $|\psi\rangle$ and an auxillary state $|A\rangle$. It is not possible to have a *universal deleter* U which is a unitary operator with the following action

$$U|\psi\rangle|\psi\rangle|A\rangle = |\psi\rangle|0\rangle|A'\rangle$$

Lets try to prove this by contradiction.

$$\text{Let } |\psi\rangle = a|0\rangle + b|1\rangle$$

We then have

$$\begin{aligned} U|\psi\rangle|\psi\rangle|A\rangle &= |\psi\rangle|0\rangle|A'\rangle \\ &= (a|0\rangle + b|1\rangle)|0\rangle|A'\rangle \\ &= a|0\rangle|0\rangle|A'\rangle + b|1\rangle|0\rangle|A'\rangle \end{aligned}$$

Since U should be a linear operator we also have

$$\begin{aligned} U|\psi\rangle|\psi\rangle|A\rangle &= U(a|0\rangle + b|1\rangle)(a|0\rangle + b|1\rangle)|A\rangle \\ &= U(a^2|0\rangle|0\rangle|A\rangle + ab|0\rangle|1\rangle|A\rangle + ba|1\rangle|0\rangle|A\rangle + b^2|1\rangle|1\rangle|A\rangle) \\ &= a^2|0\rangle|0\rangle|A'\rangle + b^2|1\rangle|0\rangle|A'\rangle + Uab|0\rangle|1\rangle|A\rangle + Uba|1\rangle|0\rangle|A\rangle \end{aligned}$$

These two results are in general not equal so we have arrived at a contradiction.

> Example 8: (Measuring a Composite Quantum System) Suppose we have a generic two qubit quantum system
>
> $$|\psi\rangle = a_{ij}|i\rangle|j\rangle$$
>
> Measuring this system generalized pretty naturally from a single qubit. We have projection operators
>
> $$\Pi_{ij} = |ij\rangle\langle ij| \Rightarrow \langle\psi|\Pi_{ij}|\psi\rangle = |a_{ij}|^2$$
>
> And we can create operators on single qubits with tensor products.
>
> $$O = \Pi_0 \otimes I \Rightarrow \frac{O|\psi\rangle}{\sqrt{\langle\psi|O|\psi\rangle}} = \frac{a_{00}|00\rangle + a_{01}|01\rangle}{\sqrt{|a_{00}|^2 + |a_{01}|^2}}$$

Lets now talk about entanglement. Let $|i\rangle^A$ and $|i\rangle^B$ be qubits in two different systems. The composite system

$$|\Phi\rangle = \frac{|0\rangle^A|0\rangle^B + |1\rangle^A|1\rangle^B}{\sqrt{2}}$$

Gives us some *spooky* properties which we will explore. First notice we can't really write the above as a product $|\Phi\rangle^A|\Phi\rangle^B$ so the two systems are "entangled" in a way even though they're two different systems. We can say these two sytems share an ebit[7] which quantifies the entanglement. We can use this entanglement as a resource and use it to reason about quantum information processing. To see this lets first consider the classical case. Suppose we have two binary random variable X_A and X_B. We can define a "classically" shared entangled random bit with a probability distribution

$$p_{X_A,X_B}(x_A, x_B) = \frac{1}{2}\delta(x_A, x_B)$$

And with this there one unit of resource of a classically entangled shared bit which we can denote with $[cc]$. Similarly in the quantum case above we have one entangled bit so we have one unit of "entanglement" which we can denote with $[qq]$[8].

[7] Just like the e-boy, it really only exists in our imagination

[8] This section feels so fucking wobbly

MacKay Chapter 1: Introduction to Information Theory

Started: March 14, 2021. Finished:

We'll start by reviewing some things in discrete probability

Example 9: (The Binomial Distribution) Consider an unfair coint with probability p of coming up heads. Suppose we flip the coin N times. What is the probability distribution, mean, and variance of getting r heads? On examination we can write the probability distribution

$$P(r) = \binom{N}{r} p^r (1-p)^{N-r}$$

The mean and variance we can derive from the above probability distribution

$$\langle r \rangle = \sum_{r=0}^{N} P(r) \times r \text{ and } \operatorname{var}[r] = \langle r^2 \rangle - \langle r \rangle^2$$

From here we can use the fact that the coin tosses are independent to simplify the above calculations. First the expected value of r is the sum of the expected number heads in each coin flip (linearity of expectation.) The expected number of heads in each coin flip is $p \times 1 + (1-p) \times 0 = p$. So we can write down

$$\langle r \rangle = Np$$

Since $\langle r^2 \rangle$ has messy cross terms we won't compute that. Instead we can use the fact that variance is linear for indepent random variables and thus we only need to compute the variance for a single coin toss. In the case of $N = 1$ we have

$$\langle r^2 \rangle = p \times 1^2 + (1-p) \times 0^2 = p \Rightarrow \operatorname{var}[r]\Big|_{N=1} = p(1-p) \Rightarrow \operatorname{var}[r] = Np(1-p)$$

Definition 21: (Stirling's Approximation)

$$x! = x^x e^{-x} \sqrt{2\pi x}$$

Definition 22: (Binary Entropy Function)

$$H_2(x) = -x \log x - (1-x) \log(1-x)$$

Lets now talk about the binary symmetric channel (same things is the noisy bit-flip channel Figure 1.) We're transmitting a single bit over a channel that as probability p of getting flipped. A physical example of this includes a hard drive. If we have some harddrive that reads/writes a gigabyte per day for ten years then we're writing and reading 10^{13} bits in the harddrive's lifetime so we'd want a probability of flipping a bit $p < 10^{-13}$ if we don't want any errors in the lifetime of the harddrive[9] To improve the rate of error we could add a encoder that encodes

[9] MacKay says 10^{-15} and I'm not sure where he got that number from. Maybe I did something wrong? I multipled one gigabyte per day times a year and let mathematica handle all the units.

message to be sent through the noisy channel and a decoder that decodes messages through the noisy channel. *Information theory* deals with the theoretical limitations on a scheme like this wheras *coding theory* deals with pratical design and implementation of these coding schemes.

Example 10: (R^3 Coding Scheme for Binary Symmetric Channel) So what is a reasonable code we could use to deal with the binary symmetric channel? The most obvious choice is R^3, repeating each bit 3 times when we encode it and take a majority vote when the reciever decodes it. We want to find the probability of error for transmitting a message. Since each bit is transmitted independently we can do the analysis for a single bit. So the probability of error is

$$P(e) = P(e|s=1) + P(e|s=0) \tag{MK.1}$$

From bayes rule we can then write

$$P(e|s=i) = P(s=i)P(e,s=i) = (\text{Probability that we transmit } i) \times \begin{pmatrix}\text{Probability of an error occuring} \\ \text{given that we transmitted } i\end{pmatrix}$$

By symmetry we konw $P(e,s=1) = P(e,s=0)$ and so (MK.1) turns into

$$P(e) = P(e,s=0)(P(1)+P(0)=1) = P(e,s=0)$$

When transmitting $s=0$ there are four possible ways the noisy channel could fuck us up. That is

$$\text{Recieved} : \text{Probabtility}$$
$$111 : p^3$$
$$\text{Permutation of } 110 : 3p^2(1-p)$$

And so all together we have

$$\boxed{P(e) = P(e,s=0) = p^3 + 3p^2(1-p) = 3p^2 - 2p^3}$$

This is good and all, the probability is smaller for a range of p but it triples the cost of transmitting or storing data. To make a 1 terabyte harddrive we'd need 3 terabytes. But lets say today is just not our day and we make the unfortunate choice to keep going with this repettion code. How would R^n work?

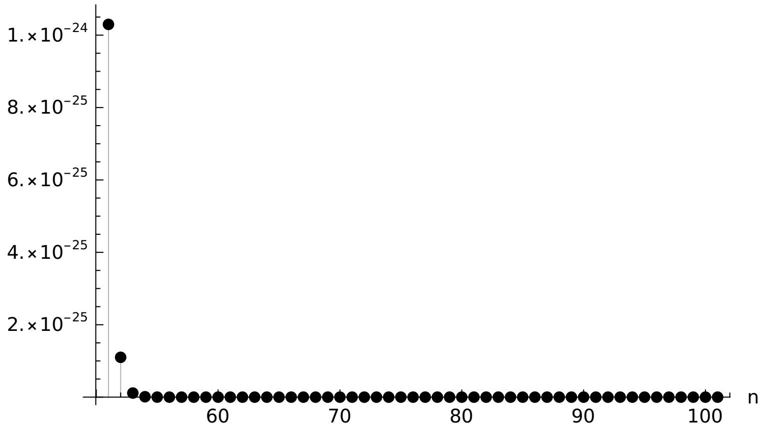
Example 11: (R^N Coding Scheme for Binary Symmetric Channel) So now when we encode the message we're repeating each bit N times. Remember last time we showed that

$$P(e) = P(e,s=0)$$

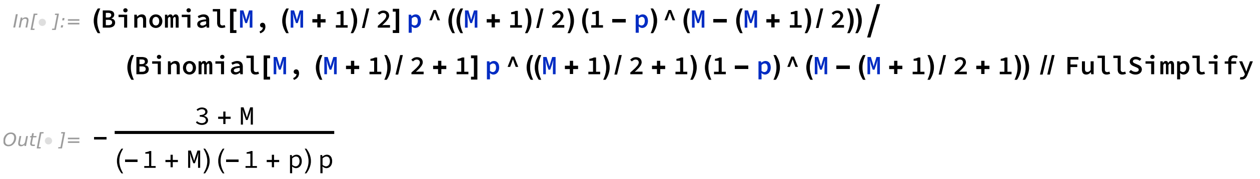
Since this result was independent of our repetition it holds here as well. Now the probability of error can be computed by simple combinatorics. In the worst case all the bits are flipped with probability p^N and there are $\binom{N}{N} = 1$ configurations of this happening. If all but one of the bits are flipped then there are $\binom{N}{N-1}$ configurations of this happening and each configuration occurs with probability $p^{N-1}(1-p)$. It shouldn't be too hard here to see a pattern. For odd N as long as more than half are flipped (e.g. more than $(N+1)/2$ bits are flipped) there is an error. From here we can immediatley write down $P(e)$.

$$P(e) = \sum_{n=(N+1)/2}^{N} \binom{N}{n} p^n (1-p)^{N-n}$$

Lets explore this space a little. Let $N = 101$ and $p = 0.1$. We can plot each term with mathematica.



So it looks like the first term is the biggest. How much bigger is the first term compared to the second term? We can do this by dividng.



```
In[∘]:= (Binomial[M, (M + 1) / 2] p ^ ((M + 1) / 2) (1 - p) ^ (M - (M + 1) / 2)) /
         (Binomial[M, (M + 1) / 2 + 1] p ^ ((M + 1) / 2 + 1) (1 - p) ^ (M - (M + 1) / 2 + 1)) // FullSimplify

Out[∘]= -      3 + M
         ─────────────────────
         (-1 + M) (-1 + p) p
```

So we can approximate the entire sum with the first term for sufficiently large N. Namely

$$P(e) \approx \binom{N}{\frac{N+1}{2}} p^{\frac{N+1}{2}} (1-p)^{(N-1)/2}$$

Now using Stirling's approximation we can simplify this even more.

$$\begin{aligned}\binom{N}{\frac{N+1}{2}} &= \frac{N!}{\left(\frac{N+1}{2}\right)!\left(\frac{N-1}{2}\right)!} \\ &\approx \frac{N^N e^{-N}}{\left(\frac{N+1}{2}\right)^{(N+1)/2}\left(\frac{N-1}{2}\right)^{(N-1)/2} e^{-N}} \\ &= \frac{(2N)^N}{(N+1)^{(N+1)/2}(N-1)^{(N-1)/2}}\end{aligned}$$

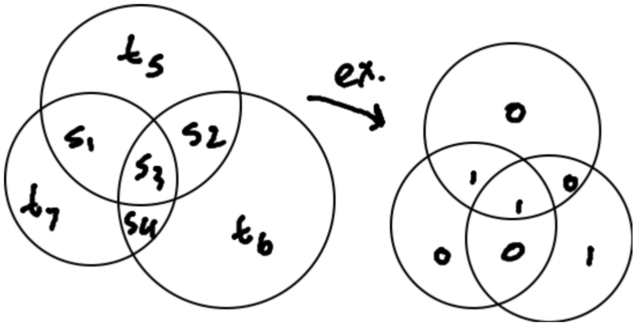
And all together we get

$$P(e) \approx (2N)^N \left(\frac{p}{(N+1)}\right)^{\frac{N+1}{2}} \left(\frac{1-p}{(N-1)}\right)^{\frac{N-1}{2}}$$

Plugging in a few values into mathemtica we need about sixty something repetitions to get below 10^{-15} probability of error. Concretley for a pretty error-free harddrive we'd need like 60 gigabytes to store 1 gigabyte (yikes.)

Hopefully you're convinced that repetition codes are kinda shit at this point. So what can we do to get better rates of information transfer? One idea is block coding. We'll encode blocks of data instead of one bit at a time. Namely we have a sequence of source bits $\mathbf{s}$ of length K that we want to apply a linear transform (in the case of a linear block code) to so that it becomes and encoded transmission vector $\mathbf{t}$ of length N. An example of this is the Hamming (7,4) code.

Definition 23: (The (7,4) Hamming Code) This is an example of a linear block code with $K = 4$ and $N = 7$. Diagramatically we can think of Venn-Diagrams where the excess bits are used to make sure the parity of the sum of bits in its domain are even.



A full enumeration of possibilities is given in MacKay's book. So for t_5 above we have $1+1+0=0$ which is even so $t_5 = 0$. Same idea for t_7. For t_6 we have $1+0+0=1$ so we need $t_6 = 1$ so that the given circle is even in parity. We can represent this code with a linear transformation $\mathbf{G}$ where

$$\mathbf{t} = \mathbf{G}^T\mathbf{s}$$

The explicit form of $\mathbf{G}^T$ is given in the book (1.26) but I'll put it here as well for my sake

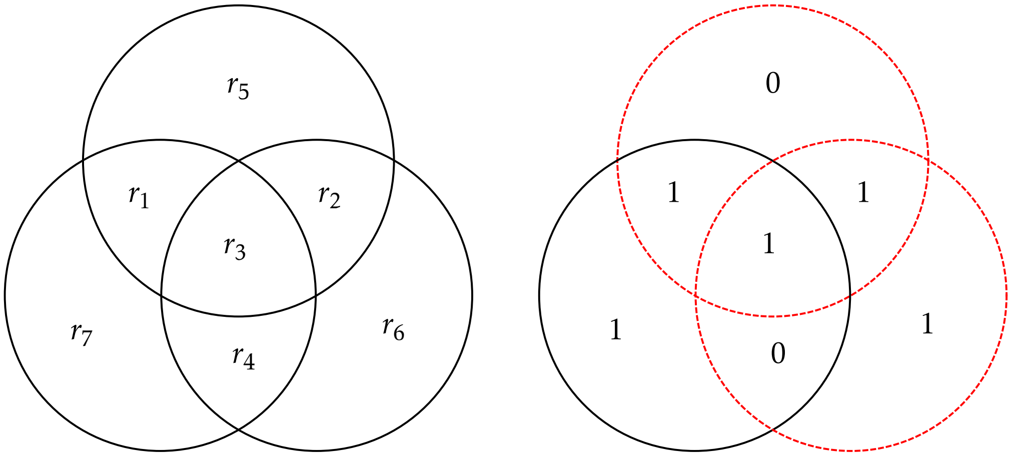
$$\mathbf{G}^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

How would we decode this fancy (7,4) Hamming code. One way is called **Syndrome decoding.**

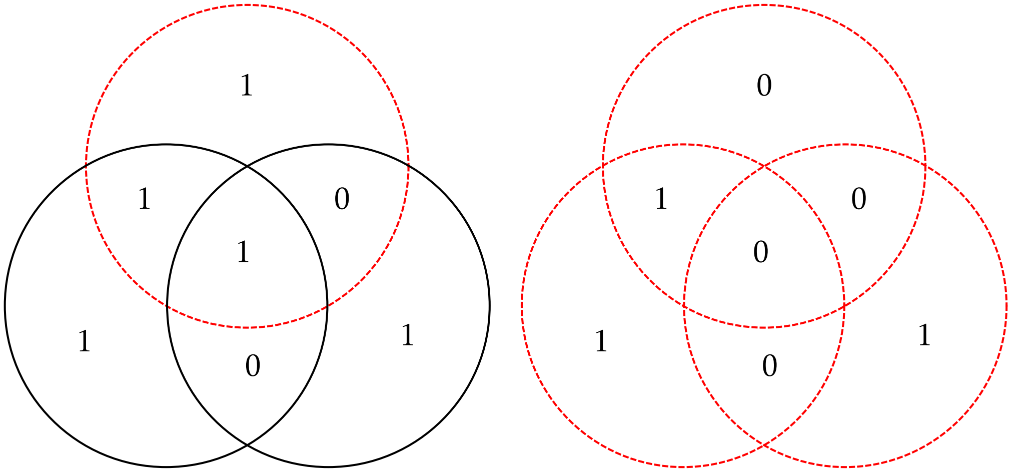
Example 12: (Syndrome Decoding for the (7,4) Hamming Code if One Bit is Flipped) The simplest case that we can analyze that's useful in introducing syndrome decoding is when one bit is flipped in the (7,4) Hamming code. Take the example we had above where $\mathbf{t} = \{1,0,1,0,0,1,0\}$ and the noise in the channell flipped the second bit. The noise vector can then be represented as $\mathbf{n} = \{0,1,0,0,0,0,0\}$ and we have the recieved signal $\mathbf{r}$ as

$$\mathbf{r} = \mathbf{t} \oplus \mathbf{n} = \{1,1,1,0,0,1,0\}$$

We can represent this diagramatically.



Where the red dotted circles are circles where the parity of the sum of bits inside that circle is not even. From this we can write down a "syndrome" which is a binary vector that encodes which circles detect an error. For the above example the syndrome $\mathbf{z}$ can be written as $\mathbf{z} = \{1,1,0\}$. Now we can figure out which bit is flipped by determining which bit lies in both the circles that detect an error while lying outside the circle that doesn't detect an error. In this case it has to be r_2. Another example is if one of the parity bits is flipped (left diagram) or the center bit was flipped (right diagram)



In these case we have $\mathbf{z} = \{1,0,0\}$ and $\mathbf{z} = \{1,1,1\}$ respectively. We can show that if only one bit is flipped then there is a well defined mapping from $\mathbf{z}$ to the flipped bit. If two bits are flipped things get tricky and errors start to happen.

We can view the (7,4) Hamming Code as a sequence of linear operators. The fist step in doing this is that we define

$$\mathbf{G}^T = \begin{bmatrix}\mathbf{I}_4 \\ \mathbf{P}\end{bmatrix}$$

$\mathbf{I}_n$ is the $n \times n$ identity matrix and $\mathbf{P}$ is what computes the parity checking bit. Now we need to find the syndrome after the message has been recieved. The syndrome is the difference of the summed bits in the domain of the parity checking bit and the actual value of the parity checking bit. With this idea in mind we can immediately write down the syndrome vector as

$$\mathbf{z} = \mathbf{H}\mathbf{r} \text{ where } \mathbf{H} = \begin{bmatrix}-\mathbf{P} & \mathbf{I}_3\end{bmatrix} \xrightarrow{\text{modulo 2 arith.}} \begin{bmatrix}\mathbf{P} & \mathbf{I}_3\end{bmatrix}$$

It should be clear that

$$\begin{bmatrix}0 \\ 0 \\ 0\end{bmatrix} = \mathbf{H}\mathbf{t}$$

We also know that $\mathbf{r} = \mathbf{G}^T\mathbf{s} + \mathbf{n}$ and by combining this with the above we see that the decoding problem reduces to finding the most probable noise vector where $\mathbf{H}\mathbf{n} = \mathbf{z}$. Lets start to analyze this coding scheme. First some definitions

Definition 24: (Probability of Block Error) The probability of block error p_B is the probability that one or more of the source bits doesn't match the recieved bits

Definition 25: (Probability of Bit Error) The probability of bit error p_b is the average probability per bit that there is an error

$$p_b = \frac{1}{K}\sum_{n=1}^{K} P(s_k \neq \hat{s}_k)$$

Example 13: (Probability of Block and Bit error of the Hamming (7,4) Code in a Binary Symmetric Channel) Lets try to analyze the Hamming (7,4) code. We want to compute the probability of a block error which occurs when two or more bits are flipped. This can be found with a simple sum and mathematica

$$P_B = \sum_{n=2}^{7}\binom{7}{n}p^n(1-p)^{7-n} = 21p^2 + O(p^3)$$

Now we want to find the leading order of the probability of bit error p_b. In this case we only need to consider when two bits are flipped since if more bits are flipped we'll get higher order terms and we only care about leading order. So if two bits are flipped what is the probability that a given bit does not equal the transmitted bit. By symmetry we know that $P(s_1 \neq \hat{s}_1)$ is the same as any other bit so lets consider s_1 in particular. In this case the probability of a bit error can be written as

$$p_b = \frac{1}{K}\times \underbrace{K\times\underbrace{\left(\underbrace{\binom{7}{2}p^2(1-p)^5}_{\substack{\text{probability that}\\ \text{2 bits flipped}}}\times P(\hat{s}_1 \neq s_1|\text{two bits flipped by noise})\right)}_{P(\hat{s}_k\neq s_k)}}_{\sum_{k=1}^{K} P(\hat{s}_k\neq s_k)}$$

And we can count use counting to determine the lilac term. What kind of noise vector would cause an error in s_1? The most obvious one would be when $\mathbf{n}$ is switched on in its first slot and thus there are 6 distinct noise vectors that would cause an error in the first bit since there are six choices for the other bit to be flipped. The other kind of noise vector that would cause a reading error in the first bit is a noise vector that leads to the syndrome $\{1,0,0\}$. Namely

$$\mathbf{Hn} = \begin{bmatrix}1\\0\\1\end{bmatrix}$$

Intuitivley this would consist of a set of noise vectors that are disjoint from the set of noise vectors that flip the first bit explicitly(remember if two bits are flipped by the noise vector then three distinct bits in total are flipped.) We can use mathematica to check how many noise vectors satisfy the above condition

```
In[∘ ]:= H := {{1, 1, 1, 0, 1, 0, 0}, {0, 1, 1, 1, 0, 1, 0}, {1, 0, 1, 1, 0, 0, 1}}
         cnt = 0;
         For[i = 1, i ≤ 7, i ++,
          For[j = i + 1, j ≤ 7, j ++,
           cnt =
            cnt + Boole[Mod[H.Table[{Or[a == i, a == j] // Boole}, {a, 1, 7}], 2] ==
               {{1}, {0}, {1}}]]]
         cnt

Out[∘ ]= 3
```

So we see there are in total $6+3=9$ possible noise vectors that cause s_1 to be flipped given that two bits are flipped and thus

$$P(\hat{s}_1 \neq s_1|\text{two bits flipped by noise}) = \frac{9}{(\text{\# of noise vectors with two bits set})} = \frac{9}{\binom{7}{2}}$$

So plugging everything in we get

$$\boxed{p_b = 9p^2 + O(p^3)}$$

> Note: My solution for the leading order of p_b feels so janky. There's almost surely a nicer way to get this result but my brain is too *smooth* to figure that out at the moment.