Lecture 10: Noisy channels

Biology 429
Carl Bergstrom

February 13, 2008

Sources: Today, we follow Chapters 3 and 8 from Leung (2002) and Chapter 7 from Cover and Thomas (2007). Portions of what follows are quoted directly from those texts.

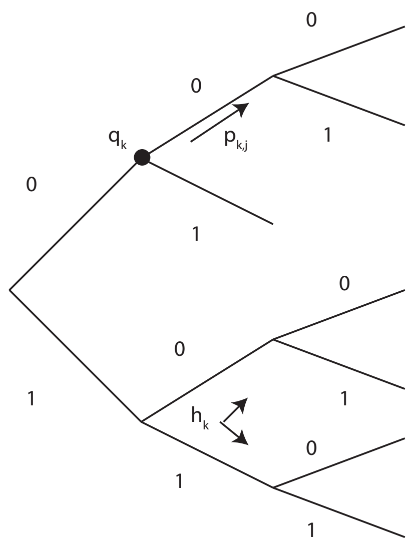
Thus far in the course, we have looked at how to send messages along channels that we have assumed to be noiseless: the message received is identical to the message sent with probability 1. Today we will relax this assumption and begin dealing with more realistic channels, namely those that are subject to noise.

Redundancy

Before proceeding, though, I want to go back and briefly treat the concept of redundancy and redundant coding. While Cover and Thomas do not treat this formally, I think it makes a useful addition to our way of thinking about noisy coding.

Definition 1 *The* redundancy *R of a uniquely decodable code C is the difference between the expected length of the code and the entropy of the source: $L(C) - H(X)$.*

We can relate redundancy of a code to the properties of the code tree. Consider a source distribution X with probabilities $\{p_1, p_2, \ldots, p_m\}$ encoded as a D-ary prefix code. The code then has a code tree with m terminal "leaves" c_i, each of which is assigned a codeword. Let $\mathcal{I}$ be the set of all internal nodes (but not leaves).



To sequentially decode a codeword, we move along the code tree, letter by letter, until we reach the terminal node.

q_k is the probability of reaching a node k during the decoding process.

$p_{k,j}$ is the probability of taking the j-th branch after reaching node k during the decoding process.

h_k is the conditional entropy at a given node. I.e., h_k is the base D entropy of the descendent probabilities $p_{k,1}, p_{k,2}, \ldots, p_{k,d}$). Notice that h_k is at most 1.

Now we can think about the process of sequentially decoding a code word using the code tree, as illustrated above.

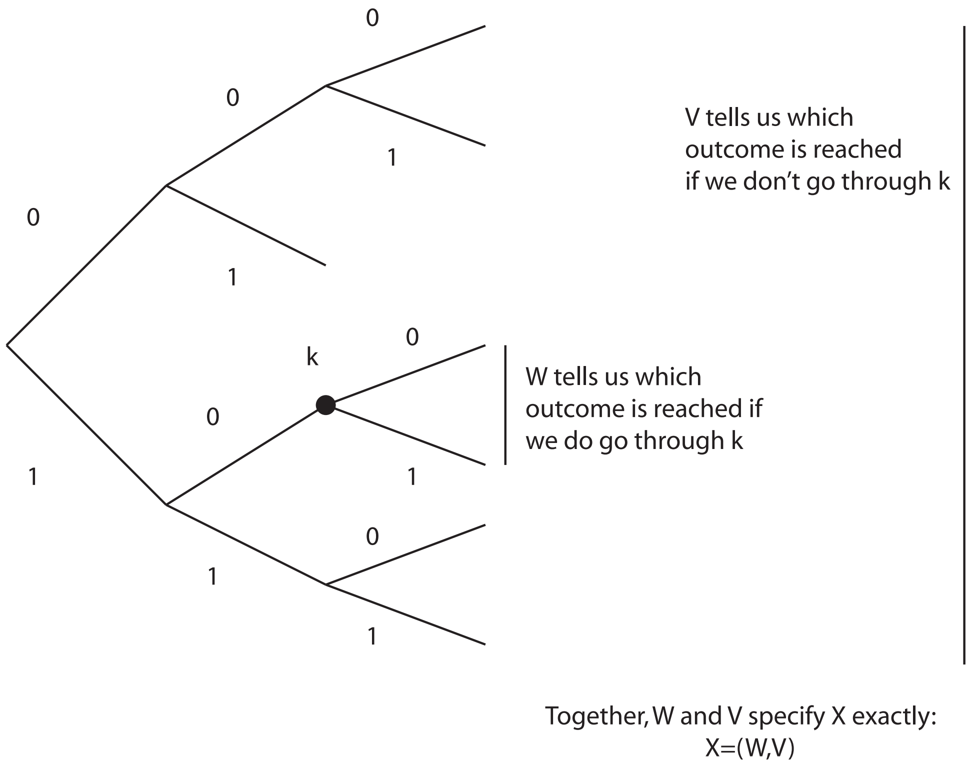
Lemma 1 *The base-D entropy of the source is equal to the sum over all internal nodes k of the reaching probability q_k times the conditional entropy at that note h_k:*

$$H_D(X) - = \sum_k q_k \, h_k.$$

The proof is by induction. For a tree with only 1 internal node (the root), the reaching probability of the node is 1 and the source entropy is by definition equal to h_1, so we've showed this for $n = 1$ nodes. Now assume that is true for n internal nodes; we want to show that it is true for $n + 1$ internal nodes.

Take any internal node k that is a parent of a leaf c_i that is all the way out at the right of the tree (i.e, $l_i = l_{\max}$). (That is, we take a node one step back into the tree.)

Now we want to break up the process of figuring out the outcome X into two parts. For part 1, we exactly identify the outcome of X if we do not travel through node k; and we simply note that we've travelled through node k — without specifying the exact outcome — if we do. We call this part the random variable V. For part 2, we exactly identify the outcome of X if we do travel through node k, and simply note that we haven't travelled through node k otherwise. We call this part the random variable W. The figure below illustrates how we break up the probabilities.


0
0
0
1
1
1
k
0
0
1
0
1
1
V tells us which outcome is reached if we don't go through k
W tells us which outcome is reached if we do go through k
Together, W and V specify X exactly: X=(W,V)

The outcome of the random variable V can be represented by a condensed code tree that prunes off the decision fork at k. This code tree has n nodes and by our induction assumption,

$$H(V) = \sum_{k' \in \mathcal{I} \setminus \{k\}} q_{k'} h_{k'}.$$

The entropy of W given V is h_k if we reach node k and 0 otherwise:

$$H(W|V) = q_k h_k + (1 - q_k) \cdot 0$$

Thus by the chain rule for entropy,

$$H(X) = H(V) + H(W|V) = \sum_{k' \in \mathcal{I}} q_{k'} h_{k'}.$$

Lemma 2 *The average code length is equal to the sum of the reaching probabilities:*

$$L = \sum_{k \in \mathcal{I}} q_k$$

Proof is straightforward by combinatorics.

Definition 2 *The* local redundancy r_k *of an internal node* k *is the product of its reaching probability and one minus its conditional entropy:*

$$r_k = q_k(1 - h_k).$$

This gives us a measure of how far away from entropy-maximizing we are at each local node or decision point. Notice that the local redundancy is zero if and only if the conditional entropy at that node is $log_D D = 1$, i.e., if that node (1) has D branches and (2) those D branches are all equi-probable.

Now we can prove a very nice theorem helping us see the relationship between local redundancy and total redundancy.

Theorem 1 *Local redundancy theorem. The total redundancy of a prefix code* C *is simply the sum of the local redundancies at each internal node of the code tree:*

$$R = \sum_{k \in \mathcal{I}} r_k.$$

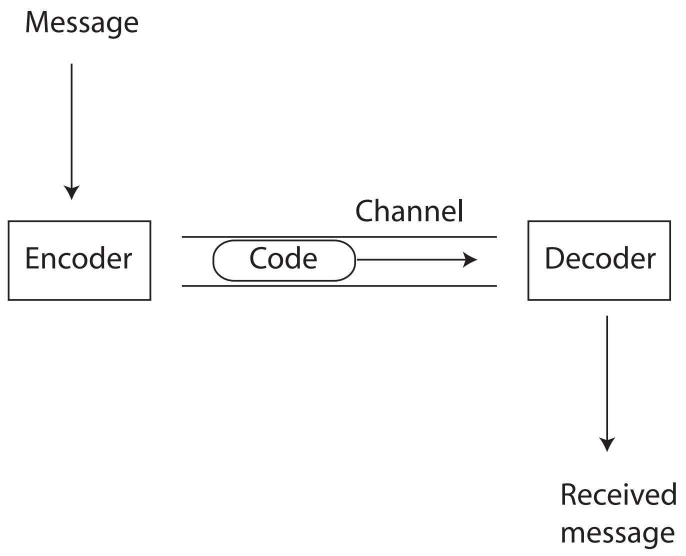
Proof: By definition, $R = L - H_D(X)$. By our two lemmas, we can replace L and $H_D(X)$ as follows:

$$\begin{aligned} R &= \sum_{k \in \mathcal{I}} q_k - \sum_{k \in \mathcal{I}} q_k h_k \\ &= \sum_{k \in \mathcal{I}} q_k(1 - h_k) \\ &= \sum_{k \in \mathcal{I}} r_k. \end{aligned}$$

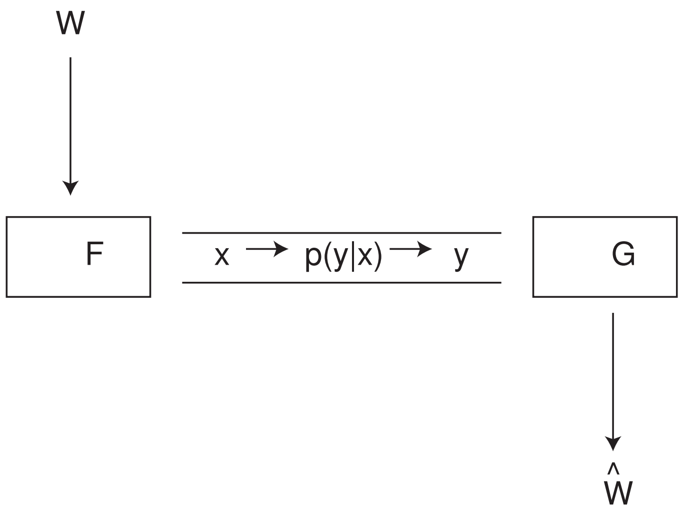
Now we can see what an efficient code tree is trying to do — it needs to minimize the redundancy at each node. In other words, each node needs to be as close to "balanced", with D branches of equal probability, as possible.

While redundancy is a bad thing in a noiseless channel, it may be useful in the sorts of noisy channels that we are about to discuss.

Noisy channels

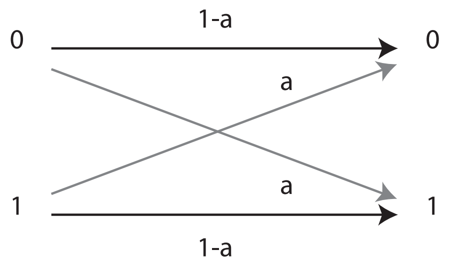


The figure above shows a schematic diagram of a typical communication channel. Some message is encoded by an encoder, sent through a channel, and comes out the other side as a received coded message with is decoded by a decoder into a received message. We let W be the original message, $\mathcal{X}$ be the channel input alphabet, and $\mathcal{Y}$ be the channel output alphabet. For many channels, $\mathcal{X}$ and $\mathcal{Y}$ will be the same. The probability of observing symbol y at the decoder given each input symbol x at the encoder is $p(y|x)$. The encoder is a function F that codes the message as a string of symbols, $F : W \rightarrow X^n$, and the decoder is a function G that decodes the received string of symbols to yield a received message, $G : Y^n \rightarrow \hat{W}$



In our previous lecture on coding, we focused only on the first step, the encoder. We assumed that the channel was noiseless: $p(y = x) = 1$ while $p(y \neq x) = 0$, and that we were working with uniquely decodable codes such that the decoder could always reverse the encoding process. Thus our aim was simply to design an encoder that could compress the message as much as possible without loss.

Today, we deal with channels that are not noiseless. Can we still encode messages in ways that we can transmit without error through the channel? Or to do that would we need to send a backup copy for each message, and a backup copy for each backup message, and so forth ad infinitum?



We begin with a model of the noise itself. As usual, let's start with a binary channel. With probability $1 - a > 1/2$, the channel faithfully passes the input symbol. With probability $a < 1/2$, the channel generates an error, changing the input symbol to the alternative symbol. Thus we can see that the error probability in this channel, P_e, is simply a.

We'll start with a naive approach to dealing with the noise, using a *repetition code*. The idea is simple: we want to pass a message A or a message B through the channel. To encode A, the encoder will repeat the symbol 0, a total of n times in a row where n is an odd integer. To encode B, the decoder will repeat the symbol 1, again n times in a row. The decoder will take the "majority vote" to decide how to interpret the output. Here we using a code that is highly redundant — but we are using this redundancy for a purpose: helping us deal with and correct the errors due to a noisy channel.

At first glance, this code appears to work reasonably well even if inefficiently. Suppose we are sending message A. As n gets large, by the weak law of large numbers the number of 0's approaches $n(1-a)$ in probability and the number of 1's approaches $na < n(1-a)$ in probability. Thus for any error threshold ϵ that we desire to achieve, we can choose n such that the $P_e < \epsilon$.

But here's the problem: how many bits per message does it take to achieve a highly reliable code?

Definition 3 *The (maximum achievable)* rate *of a binary code that uses n bits to encode each message M is*

$$\frac{1}{n}\log|M|$$

In other words, the rate is the log number of messages in the message set, divided by the average number of bits that it takes to encode a message. For our repetition code, the code rate is $\frac{1}{n}\log 2 = \frac{1}{n}$. which goes to zero as n gets large. Thus we cannot achieve arbitrarily low error unless we also accept arbitrarily low code rate. In other words, we can use redundancy to make the probability of error arbitrarily low, but the cost is that using a repetition code or similar, doing so takes so much redundancy that the rate of transmission becomes arbitrarily low as well.

Amazingly, there is a way to do better. We can use redundancy cleverly so that the rate of information transmission does not go to zero as the rate of error does go to zero. The central result of Shannon's original 1948 paper "A mathematical theory of communication" states that this is possible, but it doesn't actually tell us how to design a code that does so.

We'll preview the result now, but first, we need to define the *capacity* of a channel.

Definition 4 *The capacity C of a channel is given by*

$$C = \max_{p(x)} I(X,Y)$$

Now we can state the theorem.

Theorem 2 Shannon's channel coding theorem. *A discrete channel can achieve a rate R if and only if $R \leq C$, the channel capacity.*

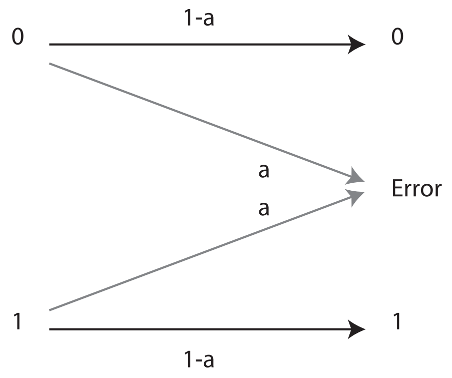
Let's look at a few examples to start to build up our intuition about how one might achieve a rate R close to capacity C. First, we compute the capacity of the symmetric binary channel modeled above.

$$\begin{aligned} I(X;Y) &= H(Y) - H(Y|X) \\ &= H(Y) - H(a) \\ &\leq 1 - H(a) \end{aligned}$$

We can achieve equality when the distribution of input symbols is uniform; thus the channel capacity $C = 1 - H(a)$.

[**Plot this**]

Next, we consider a *binary erasure channel.* In this channel, 0's and 1's are never interconverted, but with probability a they are corrupted to an error symbol E. What is the capacity of this channel?



We compute capacity as follows:

$$\begin{aligned} C &= \max_{p(x)} I(X;Y) \\ &= \max_{p(x)} H(Y) - H(a) \\ &= -H(a) + \max_{p(x)} H(Y) \end{aligned}$$

We compute $\max H(Y)$ as follows. Define a Bernoulli random variable E where $E = 1$ if the channel generates an error and $E = 0$ if the channel does not. Now

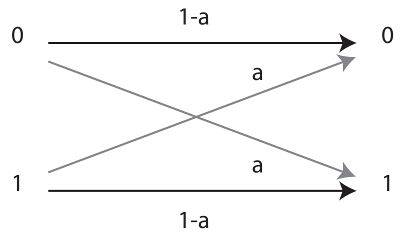
$$\begin{aligned} H(Y) &= H(Y, E) \\ &= H(E) + H(Y|E) \\ &= H(a) + (1-a)H(X) + a \cdot 0 \end{aligned}$$

Thus

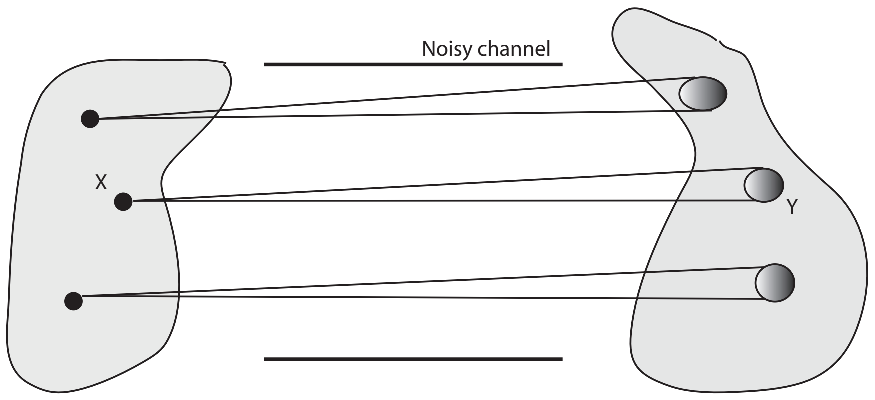
$$\begin{aligned} C &= -H(a) + \max_{p(x)} H(a) + (1-a)H(X) \\ &= (1-a) \max_{p(x)} H(X) \\ &= 1 - a \end{aligned}$$

One more example of a noisy channel, Cover's noisy typewriter. The idea is that we have a typewriter where pressing A yields either A or B as output, pressing B yields either B or C, pressing C yields C or D, etc., and pressing Z yields Z or A. We can very easily send error-free messages through this channel, by using only every other letter: A, C, E, G etc. This yields a set of outputs that can be uniquely mapped to the input, and we are thus able to send any of 13 different symbols through the channel. That is, channel capacity C is $\log 13$

This channel is a useful illustration, because it helps us understand intuitively how we can send information through a noisy channel with arbitrarily low error. The idea is that we are going to try to reconstruct a version of Cover's typewriter, choosing input codewords that may be changed somewhat as they go through the channel, but that are arbitrarily unlikely to be changed enough that they can be confused with one another.



For our symmetric binary channel, we obviously cannot do this with codewords of length 1; these are not arbitrarily unlikely to be changed into one other. But if we take our codewords to be long strings of symbols (0 or 1), we can make it arbitrarily unlikely that one will be confused with another.



This is the logic we used when constructing the repetition code. What we need to show now, though, is that we can achieve a higher rate than the repetition code does. To do so, we will choose codewords that are (usually) typical sets in the AEP sense, and we will use what we know about the size of typical sets to show how we can obtain non-zero rates through the channel.

To do so, let us briefly revisit and then extend our earlier concept of the AEP and of typical sets. We recall that:

Theorem 3 *Asymptotic Equipartition Theorem. Let $X_1, X_2, X_3, \ldots$ be independent and identically distributed (iid) random variables drawn from some distribution with probability function $p(x)$, and let $H(X)$ be the entropy of this distribution. Then as $n \to \infty$,*

$$-\frac{1}{n} \log p(X_1, X_2, \ldots, X_n) \to H(X)$$

in probability.

Definition 5 *The typical set of sequences A_ϵ^n for a probability distribution $p(x)$ is the set of sequences $x_1, x_2, \ldots$ with average probability very close to the entropy of the random variable X:*

$$\left| -\frac{1}{n} \log p(x) - H(X) \right| \leq \epsilon$$

From this theorem and definition, it follows that:

1. The probability of any sequence in the typical set is given by the entropy rate.

2. The probability of all sequences in the typical set are the same.

3. Almost all realized sequences will be members of the typical set.

4. The typical set has approximately are $2^{nH(X)}$ members.

5. Most sequences are not weakly typical.

We now extend this to the concept of *joint typicality*; this is a notion of a typicality that relates two sequences X and Y by a probability distribution $p(y|x)$.

Definition 6 *The jointly typical set $A_\epsilon^{(n)}$ of sequences $\{(x^n, y^n)\}$ with respect to the probability distribution $p(y|x)$ is the set of length-n sequences for which x^n, y^n, and the pair (x^n, y^n) all have probabilities close to their entropies.*

$$\begin{aligned}
A_\epsilon^{(n)} = \Big\{ & (x^n, y^n) \in \mathcal{X}^n \times \mathcal{Y}^n : \\
& \left| -\frac{1}{n} \log p(x^n) - H(X) \right| < \epsilon, \\
& \left| -\frac{1}{n} \log p(y^n) - H(Y) \right| < \epsilon, \\
& \left| -\frac{1}{n} \log \left(\prod_{i=1}^{n} p(x_i, y_i) \right) - H(X, Y) \right| < \epsilon.
\end{aligned}$$

We can now prove the following theorem, which is the joint-typicality analogue of the AEP:

Theorem 4 *Let (X^n, Y^n) be length-n sequences drawn i.i.d. from $p(x^n, y^n) = \prod_{i=1}^{n} p(x_i, y_i)$. Then*

1. $Pr\left((X^n, Y^n) \in A_\epsilon^{(n)}\right) \to 1$ *as* $n \to \infty$
2. $\left|A_\epsilon^{(n)}\right| \leq 2^{n(H(X,Y)+\epsilon)}$
3. *If $(\tilde{X}^n, \tilde{Y}^n) \sim p(x^n)p(y^n)$, i.e., $\tilde{X}^n$ and $\tilde{Y}^n$ are independent with the same marginals as $p(x^n, y^n)$, then*

   $$Pr\left((\tilde{X}^n, \tilde{Y}^n) \in A_\epsilon^{(n)}\right) \leq 2^{-n(I(X;Y)-3\epsilon)}.$$

   Moreover, for large enough n, this probability gets very close to the right hand side above:

   $$Pr\left((\tilde{X}^n, \tilde{Y}^n) \in A_\epsilon^{(n)}\right) \geq (1-\epsilon)2^{-n(I(X;Y)+3\epsilon)}.$$

To prove this, we begin by simply replicating our proof of the AEP, this time taking $\epsilon/3$ instead of ϵ as our bound. This tells us that there exist n_1, n_2, and n_3 such that for all $n > n_i$,

$$\Pr\left(\left|-\frac{1}{n}\log p(X^n) - H(X)\right| < \epsilon\right) \leq \frac{\epsilon}{3}$$

and

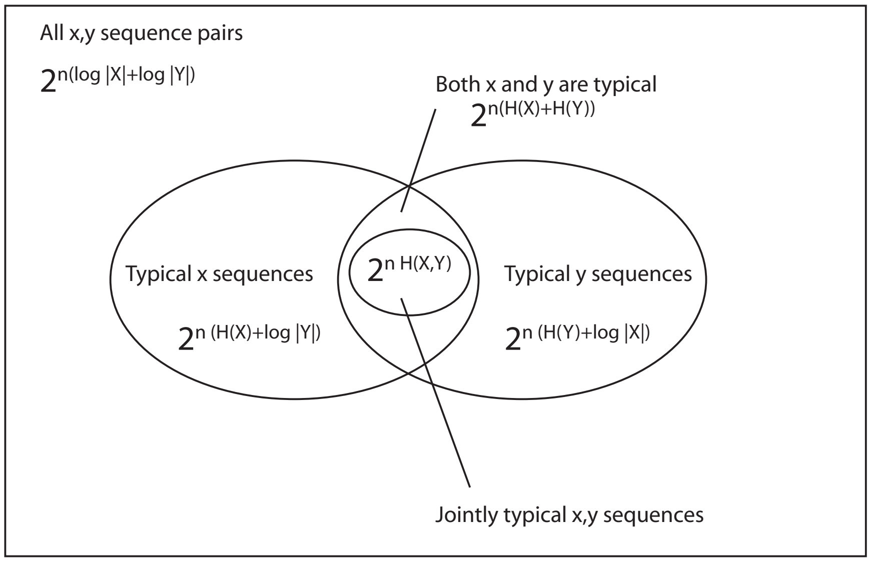
$$\Pr\left(\left|-\frac{1}{n}\log p(Y^n) - H(Y)\right| < \epsilon\right) \leq \frac{\epsilon}{3}$$

and

$$\Pr\left(\left|-\frac{1}{n}\log p(X^n, Y^n) - H(X,Y)\right| < \epsilon\right) \leq \frac{\epsilon}{3}$$

Now we take $n = \max\{n_1, n_2, n_3\}$. Then we have at least probability $1-\epsilon$ that we land in set A_ϵ^n, proving item (1). The proofs of (2) and (3) are provided in Cover and Thomas pp.196–197.

We can view, conceptually, the space of all possible sequences and see where the typical and jointly typical sets lie.



Jointly typical pairs (x^n, y^n) are relatively rare even when both x^n and y^n are themselves typical; in fact we can prove that the probability that a randomly chosen pair of typical sequences is jointly typical will be $2^{-nI(X;Y)}$. The proof can be sketched as follows. There are roughly $2^{nH(X)}$ typical X sequences and $2^{nH(Y)}$ typical Y sequences; all are equally likely by the AEP. Thus there are $2^{n(H(x)+H(Y))}$ pairs, all equally likely. Of these, we know that $2^{nH(X,Y)}$ are jointly typical, by the joint AEP, again all equally likely. Thus the probability that we pick a jointly typical pair when randomly drawing two typical sequences is approximately

$$\frac{2^{nH(X,Y)}}{2^{n(H(X)+H(Y))}} = 2^{nH(X,Y)-H(X)-H(Y))} = 2^{-nI(X;Y)}$$

Thus we can construct roughly $2^{nI(X;Y)}$ uniquely distinguishable signals X^n. This is the intuition behind Shannon's theorem, and why the rate R can approach but not exceed the channel capacity $C = \max_{p(x)} I(X;Y)$.

Recall from the last time the theorem itself:

Theorem 5 Shannon's channel coding theorem. *A discrete channel can achieve a rate R if and only if $R \leq C$, the channel capacity.*

Today let's prove that we can achieve capacity.

We generate a code *at random* — this is just a trick to make the proof symmetrical and easy. We choose a set of 2^{nR} codewords picking their elements from distribution $p(x)$, i.e., we pick as our codewords 2^{nR} elements from $\mathcal{X}^n$ with probabilities $\prod_{i=1}^{n} p(x_i)$. Both signaller and receiver know the code.

We pick a message W uniformly from the set of 2^{nR} possible messages that we have encoded, and send this message across the channel as $X^n(W)$. The receiver gets a sequence Y^n out of our noisy channel, and guesses the message by *joint typicality decoding*: she chooses message $\hat{W}$ if and only if $(X^n(\hat{W}), Y^n)$ is jointly typical and no other message $\tilde{W}$ has a codeword that is jointly typical with Y^n. If no Y^n is jointly typical with nothing in the message set, or if it is jointly typical with multiple things in the message set, the receiver marks down an error message W_0.

The proof works as follows: we average over all possible codes and all possible messages; thus the message being sent is sent by a codeword that is simply a random draw from $\mathcal{X}^n$ with probabilities $\prod_{i=1}^{n} p(x_i)$. Errors can occur two ways, which we treat in sequence. The output might not jointly typical with the input, or the output might be jointly typical with some codeword other than the input. For n large, the probability that the output Y^n is not jointly typical with the input X^n goes to 0 by the joint AEP that we provided early today. For n large, the probability that output Y_n is itself typical also goes to 1, by the AEP. Thus the probability that the output Y^n is also jointly typical with some other input $\hat{X}^n$ goes to the probability that two randomly chosen typical sequences X^n, Y^n are also jointly typical; we've seen that this probability is $2^{-nI(X;Y)}$. Therefore, we can use on average anything below $2^{nI(X;Y)}$ input codewords and be left with an arbitrarily low probability that any output is jointly typical with a codeword other than the true input codeword. Thus we can achieve any capacity up to $C = \max_{p(x)} I(X;Y)$ through the channel with arbitrarily low probability of error.