Information Theory and Networks

Lecture 22: Error Correction Codes

Matthew Roughan
<matthew.roughan@adelaide.edu.au>
http://www.maths.adelaide.edu.au/matthew.roughan/Lecture_notes/InformationTheory/

School of Mathematical Sciences,
University of Adelaide

October 8, 2013

Part I

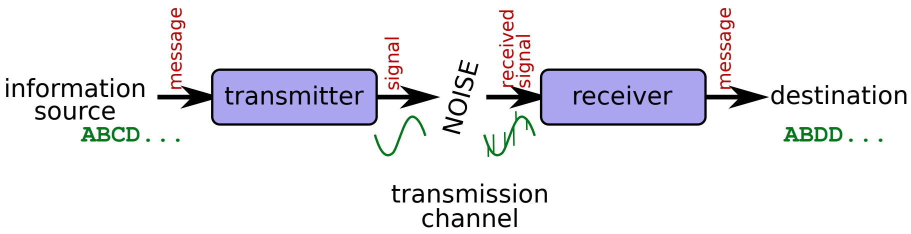
Error Correction Codes

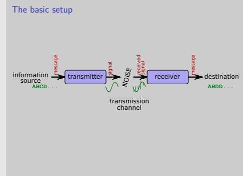
> All sorts of computer errors are now turning up. You'd be surprised to know the number of doctors who claim they are treating pregnant men.
>
> *Isaac Asimov*

Section 1

The Noisy Channel

The basic setup





An example of the transmission channel consider PSK (Phase Shift Keying).

- Phase is used to indicate a symbol
  - any sine wave has amplitude, frequency, and phase
  - phase shifts can be WRT a reference signal (Coherent PSK) or WRT to the signal itself (Differential PSK)
- Represent phase as a point on the complex plane
  - each symbol is a point on the unit circle
- Example
  - Binary-PSK (BPSK) - two phases
  - Quadrature-PSK (QPSK) - four phases
- Alternatives
  - ASK - Amplitude Shift Keying
  - FSK - Frequency Shift Keying

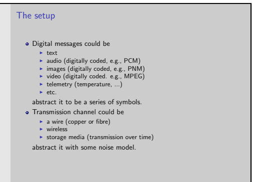
PSK is comparatively simple (to build), and so is used often, e.g. in wireless LAN standards, RFID, ...

The setup

- Digital messages could be
  - text
  - audio (digitally coded, e.g., PCM)
  - images (digitally coded, e.g., PNM)
  - video (digitally coded. e.g., MPEG)
  - telemetry (temperature, ...)
  - etc.

  abstract it to be a series of symbols.
- Transmission channel could be
  - a wire (copper or fibre)
  - wireless
  - storage media (transmission over time)

  abstract it with some noise model.

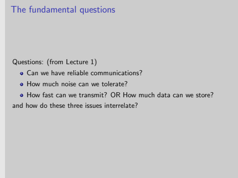


The fundamental questions

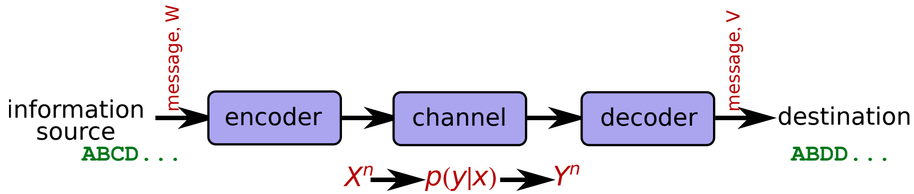
Questions: (from Lecture 1)

- Can we have reliable communications?
- How much noise can we tolerate?
- How fast can we transmit? OR How much data can we store?

and how do these three issues interrelate?



Digital Communications Channels

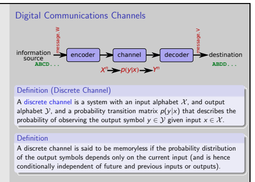


Definition (Discrete Channel)

A discrete channel is a system with an input alphabet $\mathcal{X}$, and output alphabet $\mathcal{Y}$, and a probability transition matrix $p(y|x)$ that describes the probability of observing the output symbol $y \in \mathcal{Y}$ given input $x \in \mathcal{X}$.

Definition

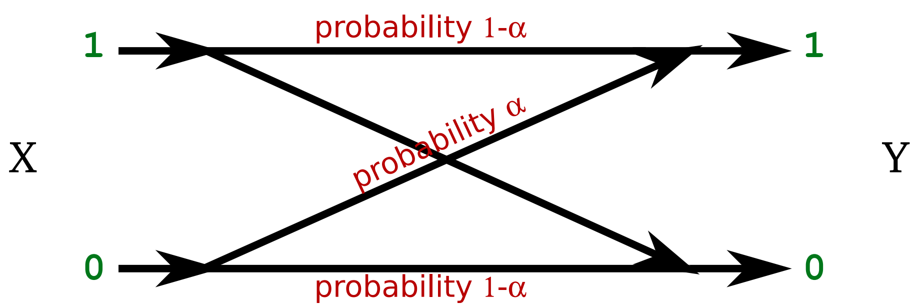
A discrete channel is said to be memoryless if the probability distribution of the output symbols depends only on the current input (and is hence conditionally independent of future and previous inputs or outputs).



$p(y|x)$ is a little like the transition matrix in a Markov chain, but

1. the input and output states don't have to be the same set
2. we only go through one step, so there is no "chain"

Example 1: Binary Symmetric Channel

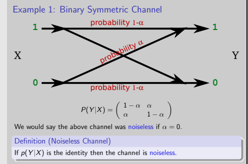


$$P(Y|X) = \begin{pmatrix} 1-\alpha & \alpha \\ \alpha & 1-\alpha \end{pmatrix}$$

We would say the above channel was noiseless if $\alpha = 0$.

Definition (Noiseless Channel)

If $p(Y|X)$ is the identity then the channel is noiseless.



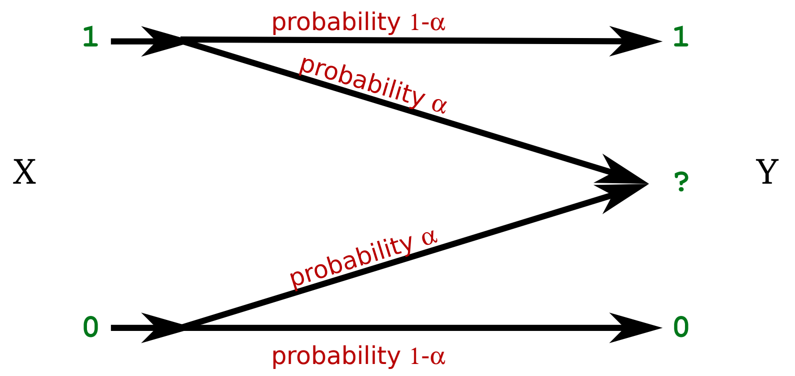
[CT91, 8.1.4, pp.186-187] or [Mac11, p.148] or [?, p.124].

$$\begin{aligned} p(Y=0 \mid X=0) &= 1-\alpha \\ p(Y=1 \mid X=0) &= \alpha \\ p(Y=0 \mid X=1) &= \alpha \\ p(Y=1 \mid X=1) &= 1-\alpha \end{aligned}$$

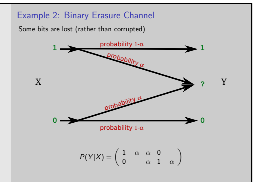
This case is obviously a discrete, memoryless channel.

Example 2: Binary Erasure Channel

Some bits are lost (rather than corrupted)



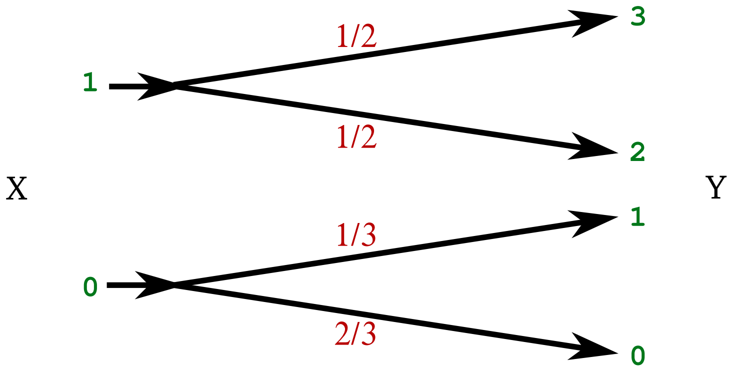
$$P(Y|X) = \begin{pmatrix} 1-\alpha & \alpha & 0 \\ 0 & \alpha & 1-\alpha \end{pmatrix}$$



[CT91, 8.1.5, pp.187-188] or [Mac11, p.148] or [?, p.124].

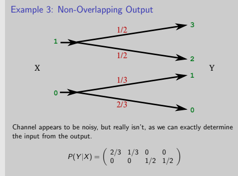
$$\begin{aligned} p(Y=0 \mid X=0) &= 1-\alpha \\ p(Y=? \mid X=0) &= \alpha \\ p(Y=1 \mid X=0) &= 0 \\ p(Y=0 \mid X=1) &= 0 \\ p(Y=? \mid X=1) &= \alpha \\ p(Y=1 \mid X=1) &= 1-\alpha \end{aligned}$$

Example 3: Non-Overlapping Output



Channel appears to be noisy, but really isn't, as we can exactly determine the input from the output.

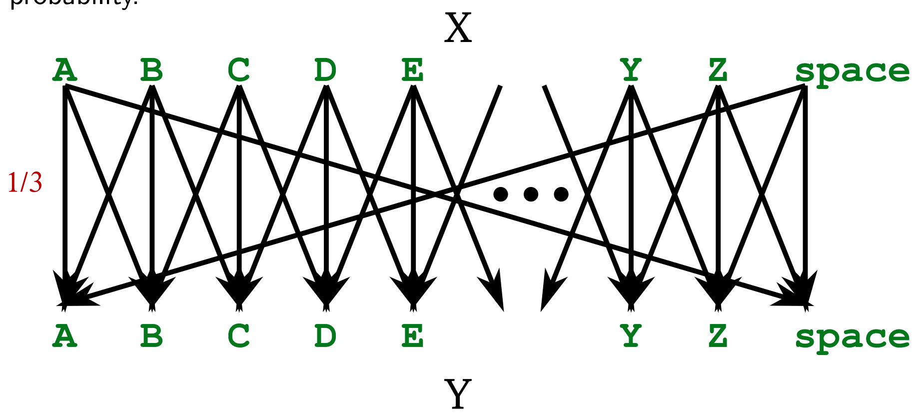
$$P(Y|X) = \begin{pmatrix} 2/3 & 1/3 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 \end{pmatrix}$$

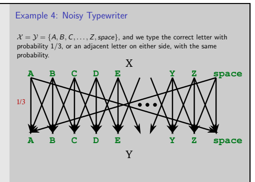


[CT91, 8.1.2, pp.185-186] or [Mac11, p.148].

Example 4: Noisy Typewriter

$\mathcal{X} = \mathcal{Y} = \{A, B, C, \ldots, Z, space\}$, and we type the correct letter with probability $1/3$, or an adjacent letter on either side, with the same probability.

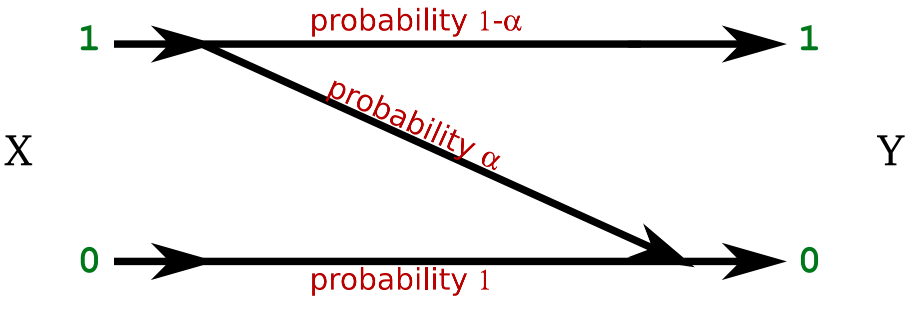




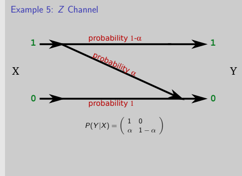
[CT91, 8.1.3, pp.185-186] or [Mac11, p.148].

$$P(Y|X) = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & \cdots & 0 & 0 & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & \cdots & 0 & 0 & 0 \\ & & \ddots & \ddots & \ddots & \ddots & & & \\ & & & \ddots & \ddots & \ddots & \ddots & & \\ 0 & 0 & 0 & 0 & 0 & \cdots & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 & 0 & 0 & 0 & \cdots & 0 & \frac{1}{3} & \frac{1}{3} \end{pmatrix}$$

Example 5: Z Channel



$$P(Y|X) = \begin{pmatrix} 1 & 0 \\ \alpha & 1-\alpha \end{pmatrix}$$



[Mac11, p.148] or [?, p.124].

$$\begin{aligned}
p(Y=0 \mid X=0) &= 1 \\
p(Y=1 \mid X=0) &= 0 \\
p(Y=0 \mid X=1) &= \alpha \\
p(Y=1 \mid X=1) &= 1-\alpha
\end{aligned}$$

The z channel is used to model some data storage systems, where only one type of error is possible:
http://en.wikipedia.org/wiki/Z-channel_(information_theory)

Native ternary codes

- Most codes are actually binary (even if they don't look it)
  - e.g. ASCII makes Letters into binary
  - binary works well with current digital computer systems
  - various attempts at ternary, or other computers rarely seem to work
- One example of native ternary computation is in a TCAM
  - CAM = Content Addressable Memory
    - useful for lots of computation tasks
    - instead of providing, say an array full of data, and having to search it for a particular value (to get the index), you can directly look up the content-value pair in one operation
    - think of as hardware associative array
  - TCAM = Ternary Content Addressable Memory
    - can match contents against 1 or 0 or ? (don't care)
    - useful for routers looking up Internet packet addresses

      1?110 matches 11110 or 10110

  - only used for specialised tasks as they are expensive (cost and energy)

http://en.wikipedia.org/wiki/Content-addressable_memory
http://en.wikipedia.org/wiki/Three-valued_logic
http://en.wikipedia.org/wiki/Ternary_computer

Section 2

Channel Capacity

Section 2
Channel Capacity

Channel Capacity

Definition (Operational Channel Capacity)

The highest rate of bits we can send per input symbol, with an arbitrarily low probability of error is called the operational channel capacity.

Let's try and work on what this could be.

Channel Capacity
Definition (Operational Channel Capacity)
The highest rate of bits we can send per input symbol, with an arbitrarily low probability of error is called the operational channel capacity.
Let's try and work on what this could be.

Channel Capacity: Q0

How much information could we theoretically pump through a channel?

Let's start with units

- Channel capacity C is measured in **bits / input symbol**
  - Remember, with D-ary code, the there are D symbols we can send.
  - With no errors, and fixed length binary codes we have $log_2 D$ bits per symbol
    - e.g., ASCII: there are 256 symbols (the numbers 0-255) with 8 bits per symbol
- Transmission rate T **symbols / second**
- Channel Rate $= C \times T$ **bits / second**
  - commonly there is a tradeoff
  - e.g. the following are equivalent
    - 256 symbols @ 1 per second OR 2 symbols @ 8 per second

Channel Capacity: Q0
How much information could we theoretically pump through a channel?
Let's start with units
Channel capacity C is measured in bits / input symbol
Remember, with D-ary code, the there are D symbols we can send.
With no errors, and fixed length binary codes we have $log_2 D$ bits per symbol
e.g., ASCII: there are 256 symbols (the numbers 0-255) with 8 bits per symbol
Transmission rate T symbols / second
Channel Rate $= C \times T$ bits / second
commonly there is a tradeoff
e.g. the following are equivalent
256 symbols @ 1 per second OR 2 symbols @ 8 per second

Channel Capacity: Q1

Does the input distribution matters?

- think about the Z-Channel example

Channel Capacity: Q1
Does the input distribution matters?
think about the Z-Channel example

Channel Capacity: Q2

What about noiseless coding/compression?

- What might be the best way to improve the rate of information?

Channel Capacity: Q2
What about noiseless coding/compression?
- What might be the best way to improve the rate of information?

Huffman Coding Example 1

X	Probability	Codeword $(z_1 z_2 \ldots z_{\ell_k})$	$p(z_i = 0 \mid X = x)$
a	0.25	01	1/2
b	0.25	10	1/2
c	0.2	11	0/2
d	0.15	000	3/3
e	0.15	001	2/3

$$
\begin{aligned}
P(z_i = 0) &= \sum_x p(z_i = 0 \mid X = x) p_X(x) \\
&= \frac{1}{2} p(X = a) + \frac{1}{2} p(X = b) + \frac{0}{2} p(X = c) + \frac{3}{3} p(X = d) + \frac{2}{3} p(X = e) \\
&= 0.5 \\
P(z_i = 1) &= 0.5
\end{aligned}
$$

Huffman Coding Example 1

Non-exact cases:

- Huffman encoding Example 2 has

$$P_Z(Z = i) = (0.57, 0.43)$$

  and

$$H(X) = 0.987 \text{ bits per symbol}$$

- Huffman encoding Example 3 (ternary code) has

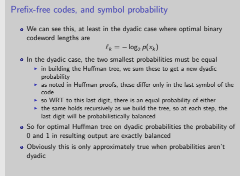
$$P_Z(Z = i) = (0.35, 0.325, 0.325)$$

  and

$$H_3(X) = 0.9994 \text{ terns per symbol}$$

Prefix-free codes, and symbol probability

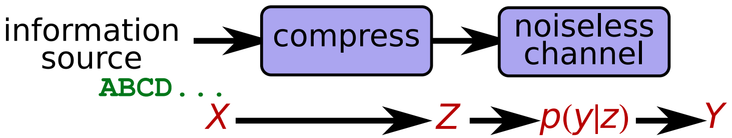
- We can see this, at least in the dyadic case where optimal binary codeword lengths are
$$\ell_k = -\log_2 p(x_k)$$
- In the dyadic case, the two smallest probabilities must be equal
  - in building the Huffman tree, we sum these to get a new dyadic probability
  - as noted in Huffman proofs, these differ only in the last symbol of the code
  - so WRT to this last digit, there is an equal probability of either
  - the same holds recursively as we build the tree, so at each step, the last digit will be probabilistically balanced
- So for optimal Huffman tree on dyadic probabilities the probability of 0 and 1 in resulting output are exactly balanced
- Obviously this is only approximately true when probabilities aren't dyadic



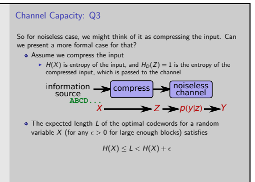
Channel Capacity: Q3

So for noiseless case, we might think of it as compressing the input. Can we present a more formal case for that?

- Assume we compress the input
  - $H(X)$ is entropy of the input, and $H_D(Z) = 1$ is the entropy of the compressed input, which is passed to the channel



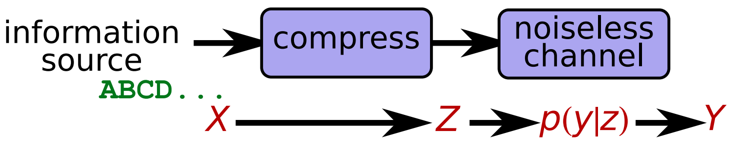
- The expected length L of the optimal codewords for a random variable X (for any $\epsilon > 0$ for large enough blocks) satisfies
$$H(X) \leq L < H(X) + \epsilon$$



L is the average codeword length
$$L = \sum p_k \ell_k$$

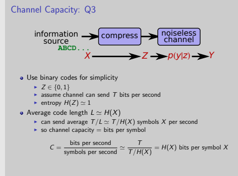
And remember that for a D-ary code, we would look at the entropy $H_D(X)$ in the result.
Also remember that by block encoding we can get L arbitrarily close to $H(X)$

Channel Capacity: Q3



- Use binary codes for simplicity
  - $Z \in \{0, 1\}$
  - assume channel can send T bits per second
  - entropy $H(Z) \simeq 1$
- Average code length $L \simeq H(X)$
  - can send average $T/L \simeq T/H(X)$ symbols X per second
  - so channel capacity = bits per symbol

$$C = \frac{\text{bits per second}}{\text{symbols per second}} \simeq \frac{T}{T/H(X)} = H(X) \text{ bits per symbol } X$$



We can think of $H(X)$ two ways

- amount of uncertainty about X before we observe it
- amount of information (in bits) we learn about X after we observe it

The second idea makes sense here: each symbols can impart $H(X)$ bits of information on the receiver.

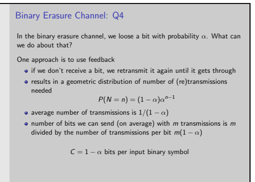
Binary Erasure Channel: Q4

In the binary erasure channel, we loose a bit with probability α. What can we do about that?

One approach is to use feedback

- if we don't receive a bit, we retransmit it again until it gets through
- results in a geometric distribution of number of (re)transmissions needed
$$P(N = n) = (1 - \alpha)\alpha^{n-1}$$
- average number of transmissions is $1/(1 - \alpha)$
- number of bits we can send (on average) with m transmissions is m divided by the number of transmissions per bit $m(1 - \alpha)$

$$C = 1 - \alpha \text{ bits per input binary symbol}$$



Can we do this without feedback?

In this example, we know when a bit is erased, but what if we have a random error that isn't obvious? One way to approach this is to do error checking using a checksum or parity bit to highlight most errors, and then retransmitting when an error occurs.

Channel Capacity: Q5

If we have noise, how would we guess the input symbol from the output?

This is a standard inference problem:

- one approach is maximum *a-posteriori* (MAP) estimation
- given output Y we estimate

$$p(X|Y) = p(Y|X)\frac{P(X)}{P(Y)} = \frac{p(Y|X)P(X)}{\sum_x P(Y|x)P(x)}$$

  by Bayes rule
- So
  - we assume we know $P(Y|X)$
  - we need to have an idea of $P(X)$
  - the bottom line is irrelevant, as it is a constant normalising factor for any given Y

  Choose the x which gives the maximum probability

What BER (Bit Error Rates) α are common? Usually expressed as power of 10, e.g., 10^{-9} would mean one bit in a billion was transmitted incorrectly.

- **optical fibre:** 10^{-10} – 10^{-14}
  - Recommendation of 10^{-13} BER for 10Gb Ethernet
- **Cat-5(e) twisted pair:** 10^{-10} – 10^{-14}
- **wireless:** 10^{-2} – 10^{-6}
  - IEEE 802.11(a,b) receiver should have min BER of 10^{-5}

Does depend on the D2A coding, conditions, and bit rates.

Often estimated by looking at the noise, and inferring, rather than direct estimation simply because the numbers may be too small.

In reality bit errors can be correlated, though we often assume that a channel is memoryless.

Example: Binary Symmetric Channel

Take, for example, output $Y = 1$

$$\begin{aligned}\operatorname*{argmax}_x p(X = x|Y = 1) &= \operatorname*{argmax}_x p(Y = 1|X = x)P(X = x) \\ &= \operatorname*{argmax}_x \{\alpha P(X = 0), (1-\alpha)P(X = 1)\}\end{aligned}$$

- assume the errors aren't too big
- assume that the input probabilities aren't too unbalanced

Then this will return $X = 1$, and similarly $X = 0$ when $Y = 0$.

- error rate is α
- with error checking and retransmission, this would look like the binary erasure channel, but the amount of checking creates an overhead, which we need to estimate

Fano's Inequality

Suppose we know a random variable Y and want to guess the value of X

- We want a function $g(Y) = \hat{X}$
- Obviously:
  - Y only helps guess X if $H(X|Y) > 0$
  - When X is completely dependent on Y, its easy to guess

  so we can see the $H(X|Y)$ is important for this problem.
- Fano's inequality formalises the question

Theorem (Fano's Inequality)

Given RVs X and Y related by $p(y|x)$, and an estimate $\hat{X} = g(Y)$, of X based on Y with probability of error $P_e = P(X \neq \hat{X})$ then

$$H(P_e) + P_e \log\left(|\mathcal{X}| - 1\right) \geq H(X|Y)$$

The inequality can be weakened, and rearranged to give

$$P_e \geq \frac{H(X|Y) - 1}{\log|\mathcal{X}|}$$

Note that $P_e = 0$ implies that $H(X|Y) = 0$, as our intuition suggested.

Note also that when we have a binary code $|\mathcal{X}| = 2$, and so the 1st inequality becomes:

$$H(P_e) \geq H(X|Y)$$

and the second is

$$P_e \geq H(X|Y) - 1$$

so there is a direct relationship between errors and the conditional entropy.

The proof of Fano's inequality is in [CT91, p.39-40] along with an example showing the inequality is sharp.

Channel Capacity: Q6

Is entropy relevant here?

- $H(X)$ is the uncertainty about X before we know the output
  - we know this was important for noiseless channels
- $H(X|Y)$ is the information we gain about X from Y
  - if this is small (e.g. they are nearly completely dependent), then there is little noise, and small errors
  - if this is nearly $= H(X)$ (e.g., they are almost independent), and we learn little about inputs from the outputs
- So why not think of capacity something like?

$$C = H(X) - H(X|Y)$$

which is the amount of information we learn about X by observing Y.

Duality

Duality between data compression and error correction:

- Redundancy in language exists, at least in part, to help correct potential errors
  - redundancy helps because some sequences are impossible or unlikely
  - we can look for a "nearby" one that is more likely
- In compression, we were coding to remove this redundancy
- To do error correction, we need to add back some redundancy
- Later, we will show that the encoder and decoder given above can be separated into two stages:
  - compression
  - error correction coding

Further reading I

Thomas M. Cover and Joy A. Thomas, *Elements of information theory*, John Wiley and Sons, 1991.

David J. MacKay, *Information theory, inference, and learning algorithms*, Cambridge University Press, 2011.