Information Theory

http://www.inf.ed.ac.uk/teaching/courses/it/

Week 7
Noisy channel coding

Iain Murray, 2012

School of Informatics, University of Edinburgh

Mutual Information revisited

Verify for yourself: $I(X;Y) = D_{\mathrm{KL}}(p(x,y)\,||\,p(x)\,p(y))$

Mutual information is non-negative:

$H(X) - H(X\,|\,Y) = I(X;Y) \geq 0,$ Proof: Gibbs' inequality

Conditioning cannot increase uncertainty on average

Concavity of Entropy

Consider $H(X) \geq H(X\,|\,C)$ for the special case:

$$p(c) = \begin{cases} \lambda & c=1 \\ 1-\lambda & c=2 \end{cases}$$

$$p(x\,|\,c{=}1) = p_1(x), \quad p(x\,|\,c{=}2) = p_2(x)$$

$$p(x) = \lambda p_1(x) + (1{-}\lambda)p_2(x)$$

which implies the entropy is concave:

$$H(\lambda\mathbf{p}_1 + (1{-}\lambda)\mathbf{p}_2) \;\;\geq\;\; \lambda H(\mathbf{p}_1) \;+\; (1{-}\lambda)H(\mathbf{p}_2)$$

Concavity of I(X;Y)

$$\begin{aligned} I(X;Y) &= H(Y) - H(Y\,|\,X) \\ &= H(Q\mathbf{p}_X) - \mathbf{p}_X^\top H(Y\,|\,x) \end{aligned}$$

First term concave in $\mathbf{p}_X$ (concave function of linear transform)

Second term linear in $\mathbf{p}_X$

Mutual Information is concave in input distribution

It turns out that $I(X;Y)$ is convex in the channel paramters Q. Reference: Cover and Thomas §2.7.

Noisy typewriter

See the fictitious noisy typewriter model, MacKay p148

For Uniform input distribution: $\mathbf{p}_X = [1/27, 1/27, \dots 1/27]^\top$

$H(X) = \log(27)$

$$p(x \,|\, y{=}\mathtt{B}) = \begin{cases} 1/3 & x = \mathtt{A} \\ 1/3 & x = \mathtt{B} \\ 1/3 & x = \mathtt{C} \\ 0 & \text{otherwise.} \end{cases} \quad \Rightarrow H(X \,|\, y{=}\mathtt{B}) = \log 3$$

$H(X \,|\, Y) = \mathbb{E}_{p(y)}[H(X \,|\, y)] = \log 3$

$I(X;Y) = H(X) - H(X \,|\, Y) = \log 27/3 = \log_2 9$ bits

Noisy Typewriter Capacity:

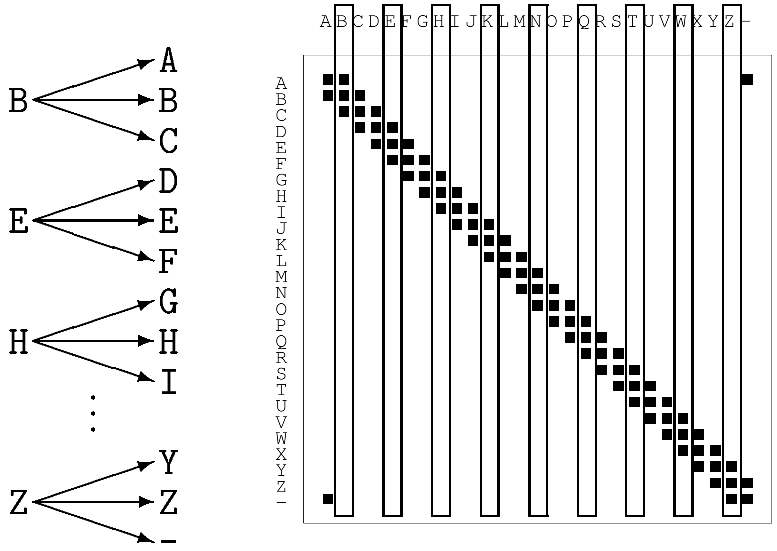
In fact, the capacity: $C = \max_{\mathbf{p}_X} I(X;Y) = \log_2 9$ bits

Proof: any asymmetric input distribution can be shifted by any number of characters to get new distributions with the same mutual information (by symmetry). Because $I(X;Y)$ is concave, any convex combination of these distributions will have performance as good or better. The uniform distribution is the average of all the shifted distributions, so can be no worse than any asymmetric distribution.

Under the uniform input distribution, the receiver infers 9 bits of information about the input. Shannon's theory will tell us that this is the fastest rate that we can communicate information without error.

For this channel there is a simple way of achieving error-less communication at this rate: only use 9 of the inputs as on the next slide. Confirm that the mutual information for this input distribution is also $\log_2 9$ bits.

Non-confusable inputs



B
A
B
C
E
D
E
F
H
G
H
I
Z
Y
Z
-
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z -
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z -

MacKay, p153

The challenge

Most channels aren't as easy-to-use as the typewriter.

How to communicate without error with messier channels?

Idea: use N^{th} extension of channel:

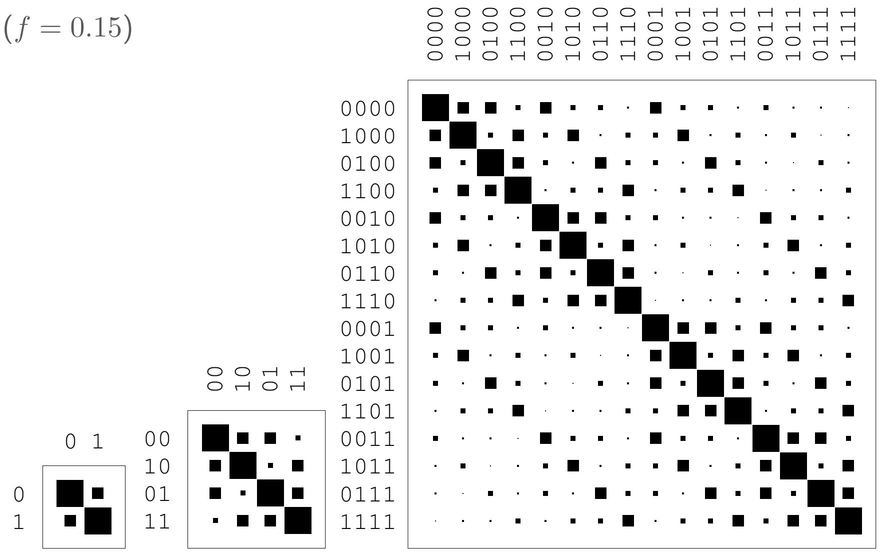
Treat N uses as one use of channel with

Input $\in \mathcal{A}_X^N$
Output $\in \mathcal{A}_Y^N$

For *large* N a subset of inputs can be non-confusable with high-probability.

Extensions of the BSC

$(f = 0.15)$

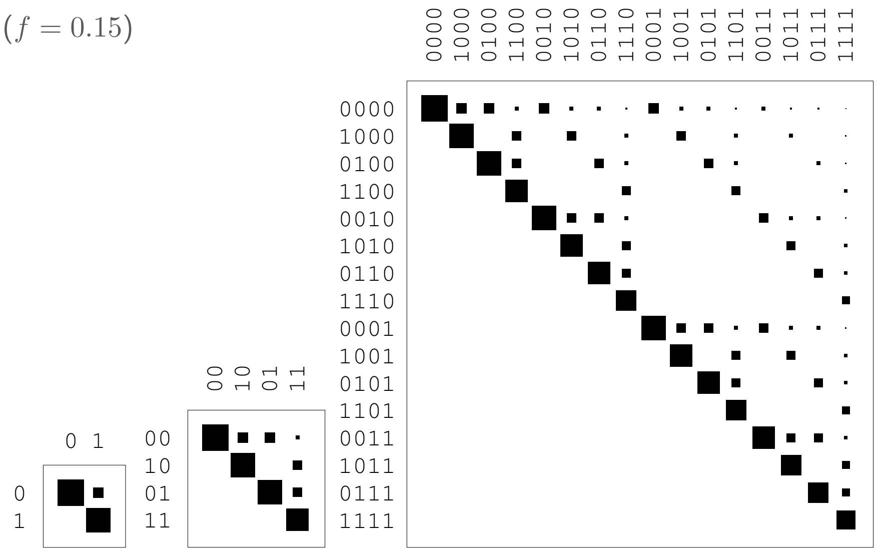


$N = 1$ $N = 2$ $N = 4$

MacKay, p153

Extensions of the Z channel

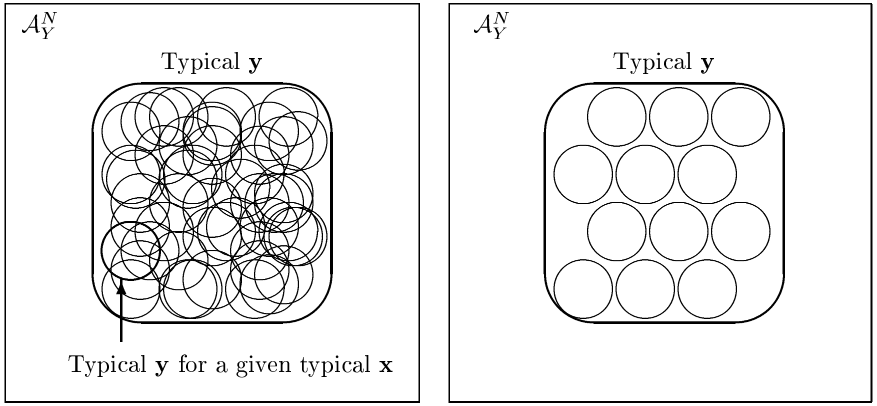
$(f = 0.15)$



$N = 1$ $N = 2$ $N = 4$

MacKay, p154

Non-confusable typical sets



MacKay, p155

Do the 4th extensions look like the noisy typewriter?

I think they look like a mess! For the BSC the least confusable inputs are `0000` and `1111` – a simple repetition code. For the Z-channel one might use more inputs if one has a moderate tolerance to error. (Might guess this: the Z-channel has higher capacity.)

To get really non-confusable inputs need to extend to larger N. Large blocks are hard to visualize. The cartoon on the previous slide is part of how the noisy channel theorem is proved.

We know from source-coding that only some large blocks under a given distribution are "typical". For a given input, only certain outputs are typical (e.g., all the blocks that are within a few bit-flips from the input). If we select only a tiny subset of inputs, *codewords*, whose typical output sets only weakly overlap. Using these nearly non-confusable inputs will be like using the noisy typewriter.

That will be the idea. But as with compression, dealing with large blocks can be impractical. So first we're going to look at some simple, practical error correcting codes.

ISBNs — checksum example

On the back of Bishop's Pattern Recognition book: (early printings)

`ISBN: 0-387-31073-8`

Group-Publisher-Title-Check

The check digit: $x_{10} = x_1 + 2\,x_2 + 3\,x_3 + \cdots + 9\,x_9 \bmod 11$

Matlab/Octave: `mod((1:9)*[0 3 8 7 3 1 0 7 3]', 11)`

Numpy: `dot([0,3,8,7,3,1,0,7,3], r_[1:10]) % 11`

Questions:

— Why is the check digit there?
— $\sum_{i=1}^{9} x_i \bmod 10$ would detect any single-digit error.
— Why is each digit pre-multiplied by i?
— Why do mod 11, which means we sometimes need `X`?

Some people often type in ISBNs. It's good to tell them of mistakes without needing a database lookup to an archive of all books.

Not only are all single-digit errors detected, but also transposition of two adjacent digits.

The back of the MacKay textbook cannot be checked using the given formula. In recent years books started to get 13-digit ISBN's. These have a different check-sum, performed modulo-10, which doesn't provide the same level of protection.

Check digits are such a good idea, they're found on *many* long numbers that people have to type in, or are unreliable to read:

— Product codes (UPC, EAN, …)
— Government issued IDs for Tax, Health, etc., the world over.
— Standard magnetic swipe cards.
— Airline tickets.
— Postal barcodes.

[7,4] Hamming Codes

Sends $K = 4$ source bits
With $N = 7$ uses of the channel

Can detect *and correct* any single-bit error in block.

My explanation in the lecture and on the board followed that in the MacKay book, p8, quite closely.

You should understand how this block code works.

To think about: how can we make a code (other than a repetition code) that can correct more than one error?

[N,K] Block codes

[7,4] Hamming code was an example of a block code

We use $S = 2^K$ codewords (hopefully hard-to-confuse)

Rate: # bits sent per channel use:

$$R = \frac{\log_2 S}{N} = \frac{K}{N}$$

Example, repetition code R_3:
$N = 3$, $S = 2$ codewords: `000` and `111`. $R = 1/3$.

Example, $[7,4]$ Hamming code: $R = 4/7$.

Some texts (not MacKay) use $(\log_{|\mathcal{A}_X|} S)/N$, the relative rate compared to a uniform distribution on the non-extended channel.
I don't use this definition.

Noisy channel coding theorem

Consider a channel with capacity $C = \max_{\mathbf{p}_X} I(X;Y)$

[E.g.'s, Tutorial 5: BSC, $C = 1-H_2(f)$; BEC $C = 1-f$]

No feed back channel

For any desired error probability $\epsilon > 0$, e.g. 10^{-15}, $10^{-30}\ldots$

For any rate $R < C$

1) There is a block code (N might be big) with error $< \epsilon$ and rate $K/N \geq R$.

2) If we transmit at a rate $R > C$ then there is a non-zero error probability that we cannot go beneath.

The minimum error probability for $R > C$ is found by "rate distortion theory", mentioned in the final lecture, but not part of this course. More detail §10.4, pp167–168, of MacKay. Much more in Cover and Thomas.

Capacity as an upper limit

It is easy to see that errorless transmission above capacity is impossible for the BSC and the BEC. It would imply we can compress any file to less than its information content.

BSC: Take a message with information content $K + NH_2(f)$ bits. Take the first K bits and create a block of length N using an error correction code for the BSC. Encode the remaining bits into N binary symbols with probability of a one being f. Add together the two blocks modulo 2. If the error correcting code can identify the 'message' and 'noise' bits, we have compressed $K + NH_2(f)$ bits into N binary symbols. Therefore, $N > K + NH_2(f) \;\Rightarrow\; K/N < 1-H_2(f)$. That is, $R < C$ for errorless communication.

BEC: we typically receive $N(1-f)$ bits, the others having been erased. If the block of N bits contained a message of K bits, and is recoverable, then $K < N(1-f)$, or we have compressed the message to less than K bits. Therefore $K/N < (1-f)$, or $R < C$.

Linear [N,K] codes

Hamming code example of linear code: $\mathbf{t} = G^{\top}\mathbf{s} \bmod 2$

Transmitted vector takes on one of 2^K codewords

Codewords satisfy $M = N-K$ constraints: $H\mathbf{t} = 0 \bmod 2$

Dimensions:

$\mathbf{t}$	$N \times 1$
$G^{\top}$	$N \times K$
$\mathbf{s}$	$K \times 1$
H	$M \times N$

For the BEC, choosing constraints H at random makes communication approach capacity for large N!

Required constraints

There are $E \approx Nf$ erasures in a block

Need E *independent* constraints to fill in erasures

H matrix provides $M = N-K$ constraints.
But they won't all be independent.

Example: two Hamming code parity checks are:

$$t_1 + t_2 + t_3 + t_5 = 0 \quad \text{and} \quad t_2 + t_3 + t_4 + t_6 = 0$$

We could specify 'another' constraint:

$$t_1 + t_4 + t_5 + t_6 = 0$$

But this is the sum (mod 2) of the first two, and provides no extra checking.

H constraints

Q. Why would we choose H with redundant rows?

A. We don't know ahead of time which bits will be erased. Only at decoding time do we set up the M equations in the E unknowns.

For H filled with $\{0, 1\}$ uniformly at random, we expect to get E independent constraints with only $M = E+2$ rows.

Recall $E \approx Nf$. For large N, if $f < M/N$ there will be enough constraints with high probability.

Errorless communication possible if
$f < (N-K)/N = 1 - R$ or if $R < 1-f$, i.e., $R < C$.

A large random linear code achieves capacity.

Details on finding independent constraints:

Imagine that while checking parity conditions, a row of H at a time, you have seen n independent constraints so far.

$P(\text{Next row of } H \text{ useful}) = 1 - 2^n/2^E = 1 - 2^{n-E}$

There are 2^E possible equations in the unknowns, but 2^n of those are combinations of the n constraints we've already seen.

Expect number of wasted rows before we see E constraints:

$$\sum_{n=0}^{E-1}\left(\frac{1}{1-2^{n-E}} - 1\right) = \sum_{n=0}^{E-1}\frac{1}{2^{E-n}-1} = 1 + {}^{1}\!/_{3} + {}^{1}\!/_{7} + \dots$$

$$< 1 + {}^{1}\!/_{2} + {}^{1}\!/_{4} + \cdots < 2$$

(The sum is actually about 1.6)

Packet erasure channel

Split a video file into $K = 10,000$ packets and transmit

Some might be lost (dropped by switch, fail checksum, . . .)

Assume receiver knows the identity of received packets:
— Transmission and reception could be synchronized
— Or large packets could have unique ID in header

If packets are 1 bit, this is the BEC.

Digital fountain methods provide cheap, easy-to-implement codes for erasure channels. They are *rateless*: no need to specify M, just keep getting packets. When slightly more than K have been received, the file can be decoded.

Digital fountain (LT) code

Packets are sprayed out continuously
Receiver grabs any $K' > K$ of them (e.g., $K' \approx 1.05K$)
Receiver knows packet IDs n, and encoding rule

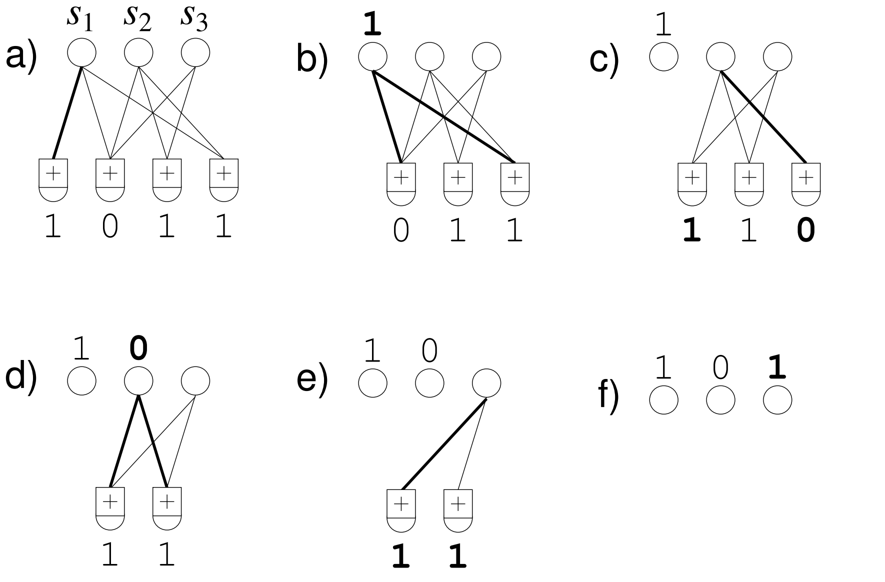
Encoding packet n:

Sample d_n pseudo-randomly from a degree distribution $\mu(d)$
Pick d_n pseudo-random source packets
Bitwise add them mod 2 and transmit result.

Decoding:

1. Find a check packet with $d_n = 1$
Use that to set corresponding source packet
Subtract known packet from all checks
Degrees of some check packets reduce by 1. GoTo 1.

LT code decoding



Soliton degree distribution

Ideal wave of decoding always has one $d=1$ node to remove

"Ideal soliton" does this in expectation:

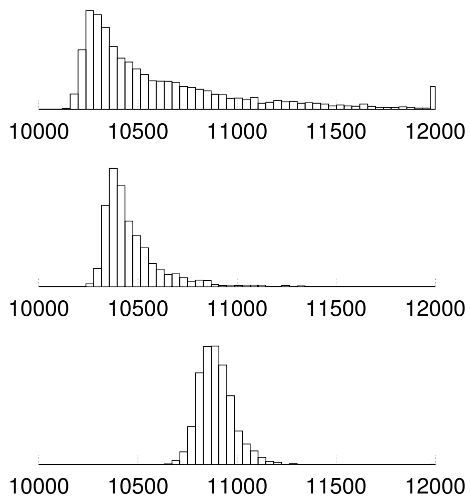
$$\rho(d) = \begin{cases} 1/K & d=1 \\ 1/d(d-1) & d=2,3,\ldots,K \end{cases}$$

(Ex. 50.2 explains how to show this.)

A robustified version, $\mu(d)$, ensures decoding doesn't stop and all packets get connected. Still get $R \to C$ for large K.

A Soliton wave was first observed in 19 C Scotland on the Union Canal.

Number of packets to catch



$K=10,000$ source packets

Numbers of transmitted packets required for decoding on random trials for three different packet distributions.

MacKay, p593

Reed–Solomon codes (sketch mention)

Widely used: e.g. CDs, DVDs, Digital TV

k message symbols $\to$ coefficients of degree $k-1$ polynomial

Evaluate polynomial at $> k$ points and send

Some points can be erased:
Can recover polynomial with any k points.

To make workable, polynomials are defined on Galois fields.

Reed–Solomon codes can be used to correct bit-flips as well as erasures: like identifying outliers when doing regression.