3F4: Data Transmission

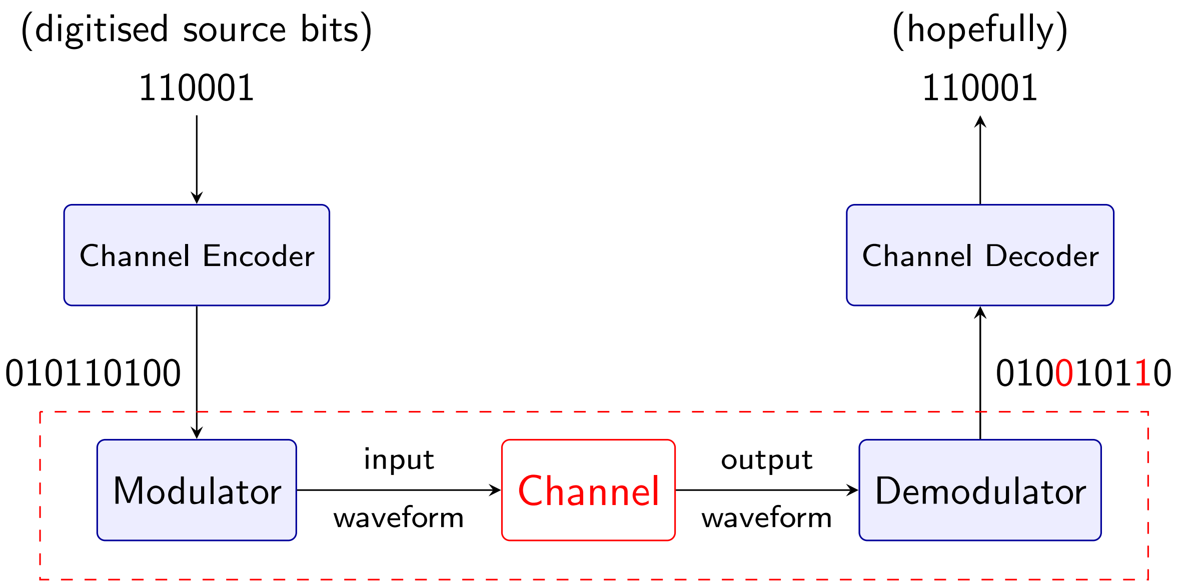
Handout 9: Channel Coding

Ioannis Kontoyiannis
[based on notes by Ramji Venkataramanan]

Signal Processing and Communications Lab
Department of Engineering
i.kontoyiannis@eng.cam.ac.uk

Lent Term 2018

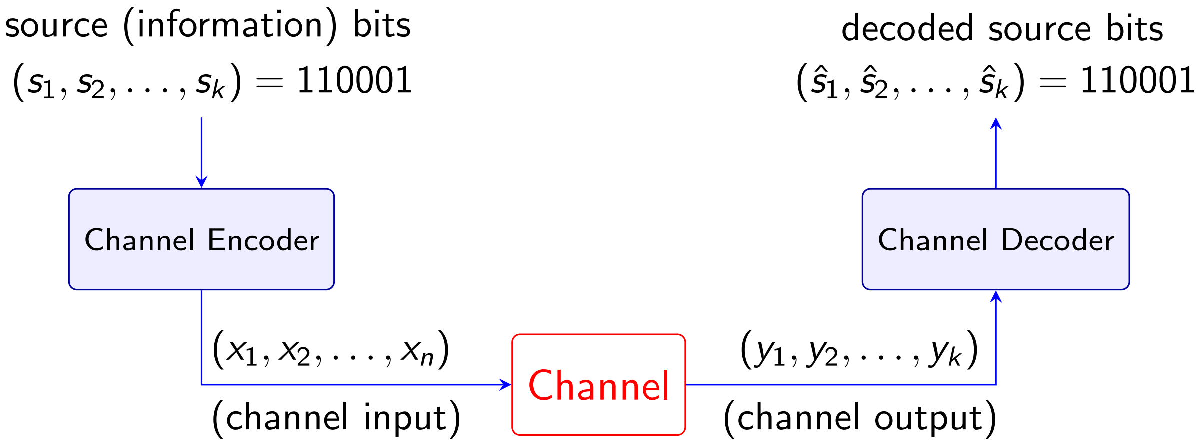
The communication channel picture so far...



(digitised source bits)
110001
Channel Encoder
010110100
Modulator
input
waveform
Channel
output
waveform
Demodulator
010010110
Channel Decoder
110001
(hopefully)

- From now on we consider the part of the system enclosed by dashed lines as **the channel**
- Focus on the design of “good” (?) coding schemes

The general channel



source (information) bits

$(s_1, s_2, \ldots, s_k) = 110001$

decoded source bits

$(\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_k) = 110001$

Channel Encoder

Channel Decoder

$(x_1, x_2, \ldots, x_n)$

(channel input)

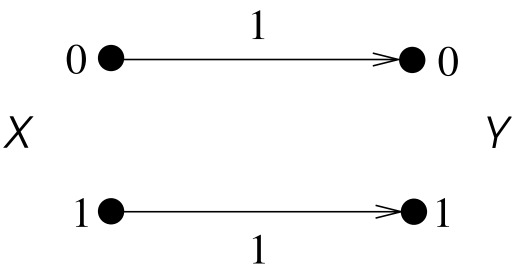
Channel

$(y_1, y_2, \ldots, y_k)$

(channel output)

- **Channel**: A collection of conditional probability distributions $P(y|x) = \Pr(Y = y|X = x)$
- **Capacity**: Fastest transmission rate, in bits per channel use, that can be achieved with arbitrarily small error probability

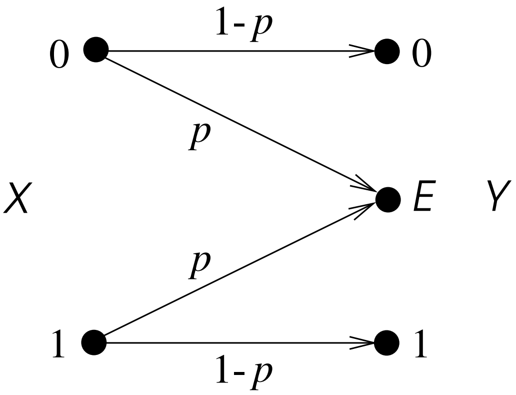
Simple channel examples



Binary noiseless channel

Capacity: Obviously

$C = 1$ bit/transmission

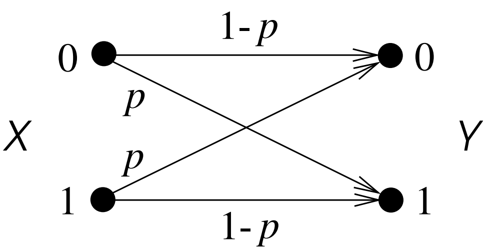


Erasure channel

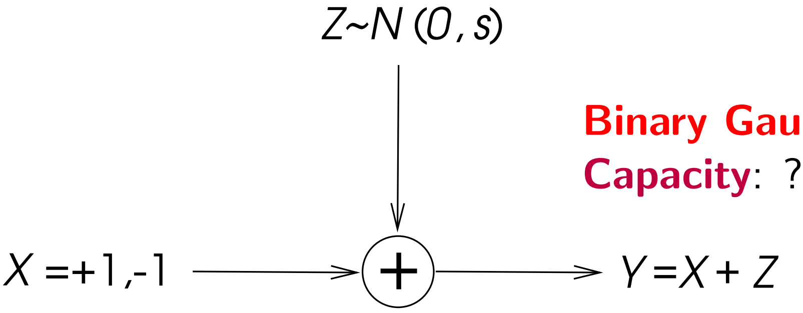
Capacity: A simple (and correct, as it turns out) guess is

$C = 1 - p$ bits/transmission

Not-so-simple channel examples

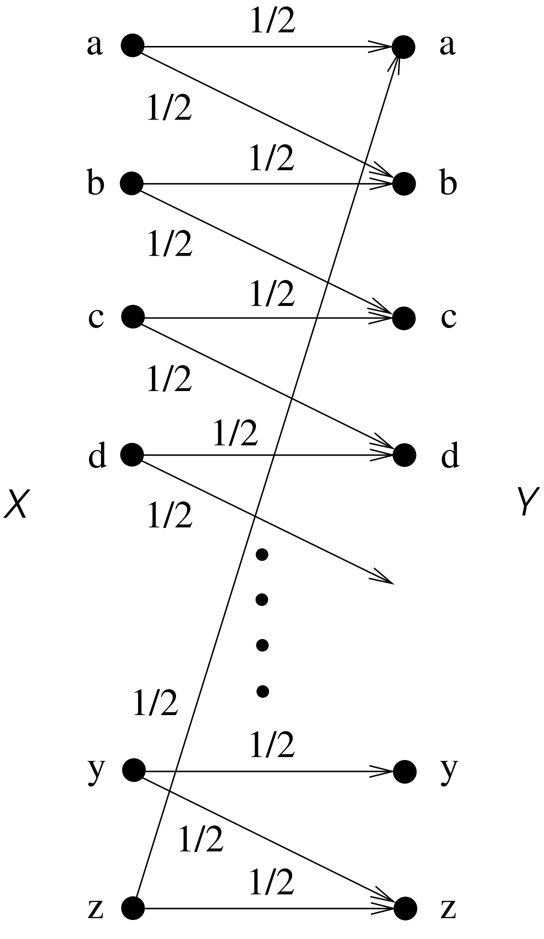


Binary symmetric channel
with crossover probability p
BSC(p) Capacity:
$C = 1 - H_2(p) < 1 - p$
bits/transmission
$[H_2(p) = -p \log_2 p - (1 - p) \log_2(1 - p)]$



Binary Gaussian noise channel
Capacity: ???

The noisy typewriter channel



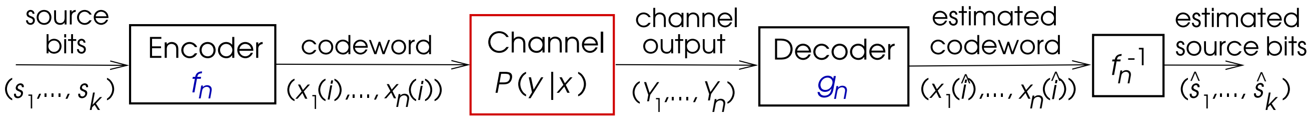
Idea: By selecting a subset $\{a, c, e, \ldots, w, y\}$ of the input symbols, we can achieve a rate of at least $\log_2 13$ bits/trans

As it turns out, the **Capacity** C is *equal* to $\log_2 13$ bits/trans

General block codes

An (n,k) **block code** maps every block of k source bits $(s_1,\dots,s_k)$ into a length n codeword $(x_1,\dots,x_n)$

- **Codebook** B_n consists of $M = |B_n| = 2^k$
  codewords $\underline{x}(i) = (x_1(i),\dots,x_n(i)),\;\; i = 1,2,\dots,M$
- **Size** of the code is $M = 2^k$ = size of codebook
- **Encoder** $f_n : \{0,1\}^k \to B_n$ maps source strings to codewords
- **Rate** of the code is $R_n = k/n$ bits/transmission
- **Channel output** string $\underline{Y} = (Y_1,\dots,Y_n) \in A_Y^n$
- **Decoder** $g_n : A_Y^n \to B_n$



source bits $(s_1,\dots,s_k)$

Encoder f_n

codeword $(x_1(i),\dots,x_n(i))$

Channel $P(y\,|\,x)$

channel output $(Y_1,\dots,Y_n)$

Decoder g_n

estimated codeword $(x_1(\hat{i}),\dots,x_n(\hat{i}))$

f_n^{-1}

estimated source bits $(\hat{s}_1,\dots,\hat{s}_k)$

Capacity

- **Error probability** (maximal) of an (n,k) code code is

$$P_e^{(n)} = \max_{1\le i\le M} \Pr(g_n(\underline{Y}) \neq \underline{x}(i)\,|\,\underline{x}(i) \text{ was sent})$$

- Rate R is **achievable** if there are (n,k) codes with $R_n \to R$ and $P_e^{(n)} \to 0$ as $n \to \infty$
- **Capacity** C = the largest achievable rate

Theorem (Shannon 1948)

As long as the input and output are not independent the capacity is positive and given by $C = \max_{P_X} I(X;Y)$

- From now on concentrate on the BSC(p) for which $C = 1 - H_2(p)$, where $H_2(p) = -p\log_2 p - (1-p)\log_2(1-p)$

Simple examples of codes

1) $(n, 1)$ *repetition code*: $0 \longrightarrow 00\cdots0 \qquad 1 \longrightarrow 11\cdots1$

2) *An* $(n = 5, k = 2)$ *block code*:

$$\begin{array}{lcl} 0\,0 & \longrightarrow & 1\,0\,1\,0\,1 \\ 0\,1 & \longrightarrow & 1\,0\,0\,1\,0 \\ 1\,0 & \longrightarrow & 0\,1\,1\,1\,0 \\ 1\,1 & \longrightarrow & 1\,1\,1\,1\,1 \end{array}$$

3) *Single parity-check* $(k + 1, k)$ *code*: Each codeword consists of the k source bits together with a $(k + 1)$th parity bit which is the modulo-two sum of the source bits

E.g., with $k = 4$, this gives a $(5, 4)$ code with 16 codewords:

$$\begin{array}{lcl} 0\,0\,0\,0 & \longrightarrow & 0\,0\,0\,0\,0 \\ 0\,0\,0\,1 & \longrightarrow & 0\,0\,0\,1\,1 \\ 0\,0\,1\,0 & \longrightarrow & 0\,0\,1\,0\,1 \\ & \vdots & \end{array}$$

Good codes

What makes a good code?

- High rate $R = k/n$
- Low probability of error $P_e^{(n)}$
- **Computationally efficient encoding and decoding**

Back to the examples

1) The $(n, 1)$ repetition code with n odd

- Decoder: If received word contains majority of zeros, declare data bit to be 0; else declare 1
- Error prob $P_e^{(n)}$ can be made arbitrarily small by taking n large
- But rate R_n is only $1/n$

3) The $(k+1, k)$ single-parity code:

- Rate $R_n = k/(k+1)$ can be made arbitrarily close to 1 by taking k large [even though the capacity $C < 1$!]
- Decoder can can always correctly *detect* one error but cannot correct it
- *Exercise*: Show that the error prob $P_e^{(n)}$ is always $\geq 1/2$ (*Hint*. What if there is an even number of channel errors?)

Minimum distance of a code

The **Hamming distance** $d(\underline{x}, \underline{y})$ between two binary sequences $\underline{x}, \underline{y}$ of equal length is the number of positions in which $\underline{x}$ and $\underline{y}$ differ.

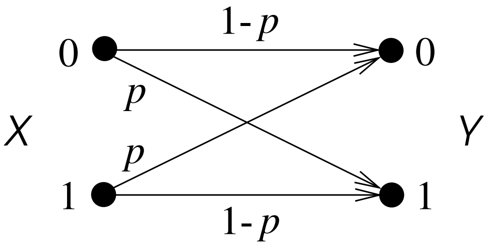
Let B_n be a codebook with codewords $\{\underline{x}(1), \ldots, \underline{x}(M)\}$. The **minimum distance** d_{min} of the corresponding code is the smallest Hamming distance between any pair of codewords

$$d_{min} = \min_{i \neq j} d(\underline{x}(i), \underline{x}(j))$$

E.g., in 2) above $d_{min} = 2$: $B_n = \{1\,0\,1\,0\,1,$
$1\,0\,0\,1\,0,$
$0\,1\,1\,1\,0,$
$1\,1\,1\,1\,1\}$

Optimal decoding of a block code for the BSC(p)

Given a block code with codebook $B_n = \{\underline{x}(1), \dots, \underline{x}(M)\}$, the optimal decoder is the one that minimises the prob of error



This is the so-called *maximum likelihood decoder* :
If $\underline{Y}$ is the channel output sequence, then we decode:

$$\hat{\underline{x}} = \arg\max_{\underline{x} \in B_n} P(\underline{Y} \mid \underline{x})$$

If codeword $\underline{x}$ is transmitted and $\underline{Y}$ is received, the number of channel errors is $d(\underline{Y}, \underline{x})$, and the number of correctly received bits in $\underline{Y}$ is $n - d(\underline{Y}, \underline{x})$. Hence:

$$P(\underline{Y} \mid \underline{x}) = p^{d(\underline{Y},\underline{x})}(1-p)^{n-d(\underline{Y},\underline{x})} = (1-p)^n \left(\frac{p}{1-p}\right)^{d(\underline{Y},\underline{x})}$$

Optimal decoding of a block code for the BSC(p)

Since $\hat{\underline{x}} = \arg\max_{\underline{x} \in B_n} P(\underline{Y} \mid \underline{x})$ and

$$P(\underline{Y} \mid \underline{x}) = p^{d(\underline{Y},\underline{x})}(1-p)^{n-d(\underline{Y},\underline{x})} = (1-p)^n \left(\frac{p}{1-p}\right)^{d(\underline{Y},\underline{x})}$$

for $p < \frac{1}{2}$ the optimal decoding rule is:

Decode $\hat{\underline{x}} = \arg\min_{\underline{x} \in B_n} d(\underline{Y}, \underline{x})$

$\Leftrightarrow$ the optimal decoder for the BSC picks the Hamming-closest codeword $\underline{x}$ to $\underline{Y}$, independently of p!

Proposition

We can successfully **correct** any pattern of $t \leq \lfloor \frac{d_{min}-1}{2} \rfloor$ errors

Linear Block Codes (LBCs)

A (n, k) *linear* block code (LBC) is defined in terms of k length-n binary vectors, say $\underline{g}_1, \dots, \underline{g}_k$. The sequence $\underline{s} = (s_1, \dots, s_k)$ of k data bits is mapped to a codeword $\underline{x} = (x_1, \dots, x_n)$ as follows:

$$\underline{x} = s_1\underline{g}_1 + s_2\underline{g}_2 + \cdots + s_k\underline{g}_k$$

This can be compactly written as

$$\underline{x} = \underline{s}\, G, \qquad \text{where } G \text{ is the } k \times n \text{ matrix } G = \begin{bmatrix} \underline{g}_1 \\ \underline{g}_2 \\ \vdots \\ \underline{g}_k \end{bmatrix}$$

- G is the **generator matrix** of the code
- k is the code **dimension**
- n is the code **blocklength**

Notation : All unspecified vectors are row vectors and all arithmetic operations are modulo 2

Example

Each of the matrices below generates a $(4, 2)$ code, of dimension $k = 2$ and blocklength $n = 4$:

$$G_1 = \begin{bmatrix} 1\,0\,0\,1 \\ 0\,1\,1\,1 \end{bmatrix} \qquad\qquad G_2 = \begin{bmatrix} 1\,1\,1\,0 \\ 0\,1\,1\,1 \end{bmatrix}$$

$\underline{s}$		$\underline{x}$	$\underline{s}$		$\underline{x}$
0 0	$\longrightarrow$	0 0 0 0	0 0	$\longrightarrow$	0 0 0 0
0 1	$\longrightarrow$	0 1 1 1	0 1	$\longrightarrow$	0 1 1 1
1 0	$\longrightarrow$	1 0 0 1	1 0	$\longrightarrow$	1 1 1 0
1 1	$\longrightarrow$	1 1 1 0	1 1	$\longrightarrow$	1 0 0 1

$\Rightarrow$ The generator matrix for a code is *not* unique

Among all possible generator matrices for a code, we often prefer one that is of the form

$$G = [\ \mathrm{I}_k \ \mid\ \mathrm{P}\ ]$$

where I_k is the $k \times k$ identity matrix and P is a $k \times (n-k)$ matrix

Systematic generator matrices

A generator matrix of the form $G = \left[\, \mathrm{I}_k \mid \mathrm{P} \,\right]$ is called a **systematic** generator matrix. Then the codeword corresponding to a length-k data string $\underline{s}$ is

$$\underline{x} = \underline{s}G = \underline{s} \cdot \left[\, \mathrm{I}_k \mid \mathrm{P} \,\right] = \left[\, \underline{s} \mid \underline{s}\,\mathrm{P} \,\right]$$

> In a systematic code, the codeword $\underline{x}$ consists of the k data bits $\underline{s}$ followed by $(n-k)$ *parity bits* $\underline{s}\,\mathrm{P}$

Example: $G_1 = \begin{bmatrix} 1\,0\,0\,1 \\ 0\,1\,1\,1 \end{bmatrix}$ is systematic. The codeword corresponding to $\underline{s} = (s_1, s_2)$ is

$$\underline{x} = [\, s_1,\ s_2,\ \underbrace{s_2,\ s_1 + s_2}_{parity\ \ bits} \,]$$

N.B. $G_2 = \begin{bmatrix} 1\,1\,1\,0 \\ 0\,1\,1\,1 \end{bmatrix}$ yields the same code, but G_2 is *not systematic*

Example: *The* $(7,4)$ *Hamming code* has blocklength $n = 7$, dimension $k = 4$, rate $R_n = 4/7$, and systematic generator matrix:

$$G = \begin{bmatrix} 1\,0\,0\,0\,1\,1\,1 \\ 0\,1\,0\,0\,1\,1\,0 \\ 0\,0\,1\,0\,0\,1\,1 \\ 0\,0\,0\,1\,1\,0\,1 \end{bmatrix}$$

Properties of LBCs

- Any generator matrix can be brought to systematic form via elementary row operations and swapping columns
- The codebook B_n of a LBC is a k-dimensional subspace of $\{0,1\}^n$, i.e., it is closed under vector addition and scalar multiplication
- The rows of G, $\{\underline{g}_1, \ldots, \underline{g}_k\}$, form a basis for the (sub-)space B_n
- The sum of any two codewords is also a codeword and the all-zero vector $\underline{0}$ is always a codeword

Minimum distance of an LBC

- The Hamming distance between $\underline{u}$ and $\underline{v}$ can be expressed

$$d(\underline{u},\underline{v}) = \text{wt}(\underline{u}+\underline{v}),$$

  where wt is the Hamming *weight* of the vector
  i.e., the number of ones in it
- Recall that the minimum distance of a code with codebook $B_n = \{\underline{x}(1), \dots, \underline{x}(M)\}$ is

$$d_{min} = \min_{i \neq j} d(\underline{x}(i), \underline{x}(j)) = \min_{i \neq j} \text{wt}(\underline{x}(i) + \underline{x}(j))$$

- Since the sum of any two codewords of an LBC is also a codeword:

$$d_{min} = \min_{i \neq j} \text{wt}(\underline{x}(i) + \underline{x}(j)) = \min_{\underline{x} \in B_n,\, \underline{x} \neq 0} \text{wt}(\underline{x})$$

The minimum distance of an LBC equals the minimum Hamming weight among the non-zero codewords.

Performance of block codes over the BSC(p)

Recall: all error patterns of weight $\leq \lfloor (d_{min}-1)/2 \rfloor$ can be corrected. Hence prob of correct decoding is at least the prob that the channel flips $\leq \lfloor (d_{min}-1)/2 \rfloor$ bits

$$\Rightarrow \quad 1 - P_e^{(n)} \geq \sum_{w=0}^{\lfloor (d_{min}-1)/2 \rfloor} \binom{n}{w} p^w (1-p)^{n-w}$$

$$\Rightarrow \quad P_e^{(n)} \leq 1 - \sum_{w=0}^{\lfloor (d_{min}-1)/2 \rfloor} \binom{n}{w} p^w (1-p)^{n-w}$$

- The above bound is too pessimistic, because d_{min} only gives the *guaranteed* error correction capability of the code
- There may be still be *many* error patterns of weight $> \lfloor (d_{min}-1)/2 \rfloor$ that can be corrected
- Early coding theory focused on constructing codes with d_{min} as large as possible. But, in general, this does not yield effective practical codes...

What do actual capacity-achieving codes look like ?

The key is to realize that we are trying to achieve a small error probability, *not zero error*

- E.g., a length $n = 10000$ code may have $d_{min} = 50$, but may be able to correct *many* error patterns of weight 1000
- With $d_{min} = 50$, there will be one or more error patterns of weight 25 that lead to decoding error, but if the total probability of these error patterns is small, their contribution to the overall error probability will be small
- What is important for good performance over a BSC(p) is that the code corrects most of the *typical* error patterns, i.e., ones with weight close to np — and not even every single one of them

Next two lectures on *convolutional codes*: A class of practical linear codes for the BSC