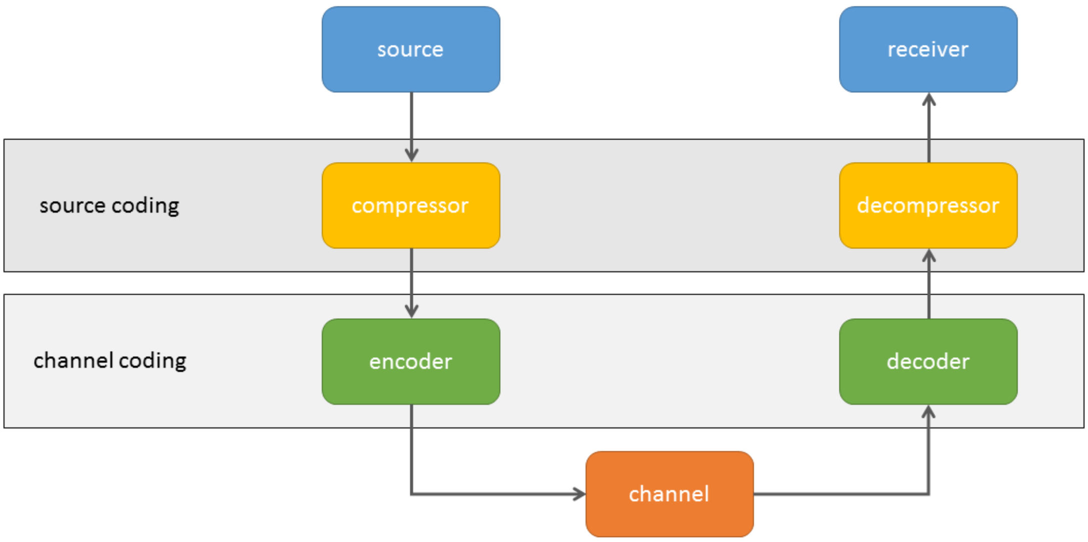
Fundamentals of Mathematical Informatics

Communication through Noisy Channels

Francesco Buscemi

Lecture Four

General communication scheme



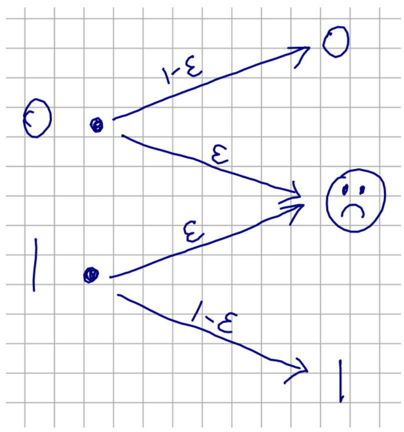
The discrete memoryless channel (DMC)

- In lecture one, we said that a RV X is like a 'device' that outputs an element from a set $\{x_1, \cdots, x_n\}$ with probability $\Pr\{X = x_i\} = p_i$.
- Imagine now a 'device' that has an output *and* an input: it accepts strings of symbols from its input alphabet $\Sigma_1 = \{a_1, \cdots, a_m\}$ and emits strings of symbols from an output alphabet $\Sigma_2 = \{b_1, \cdots, b_n\}$.
- A **discrete memoryless channel** (DMC) is given by: an input alphabet $\Sigma_1 = \{a_1, \cdots, a_m\}$, an output alphabet $\Sigma_2 = \{b_1, \cdots, b_n\}$, and a channel matrix $P = [\![p_{ij}]\!]_{ij}$ $(1 \leqslant i \leqslant m,\ 1 \leqslant j \leqslant n)$ of transition probabilities:

$$p_{ij} \stackrel{\text{def}}{=} p(b_j|a_i) \stackrel{\text{def}}{=} \Pr\{\text{output is } b_j | \text{input was } a_i\}.$$

  Therefore, $p_{ij} \geqslant 0$ for all i and j, and $\sum_j p_{ij} = 1$ for all i.
- **Memory trick.** To remember which is the input and which is the output, think as if $p_{ij} = p_{i \to j}$.
- The channel is 'discrete' because input and output alphabets are discrete sets.
- The channel is 'memoryless' because the channel matrix P remains the same for repeated uses.

Example: the binary erasure channel

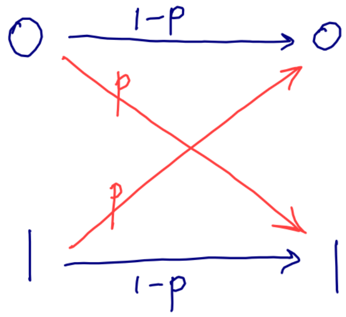


In this case, $\Sigma_1 = \{0, 1\}$, $\Sigma_2 = \{0, 1, ☹\}$, and

$P =$

	0	1	☹
0	$1 - \epsilon$	0	ϵ
1	0	$1 - \epsilon$	ϵ

Example: the binary symmetric channel



In this case, $\Sigma_1 = \Sigma_2 = \{0, 1\}$ and

$P =$

	0	1
0	$1-p$	p
1	p	$1-p$

Input and output of a DMC as RVs

- Let X be a RV with range $\mathcal{X} = \{x_1, \cdots, x_m\}$ and probability distribution $(p_1, \cdots, p_m)$.
- Take now a DMC $\mathcal{N}$ with input alphabet $\mathcal{X}$, output alphabet $\mathcal{Y} = \{y_1, \cdots, y_n\}$, and channel matrix $P = [\![p_{ij}]\!]$.
- What happens if we 'feed' X through $\mathcal{N}$?
- $\Pr\{\text{'output is } y_j\text{'}\} = \sum_{i=1}^m \Pr\{X = x_i\} p_{ij} = \sum_{i=1}^m p_i p_{ij}$.
- We obtain another RV Y, with range equal to $\mathcal{Y}$ and probability distribution $(q_1, \cdots, q_n)$ where $q_j = \sum_i p_i p_{ij}$.

I/O joint distribution

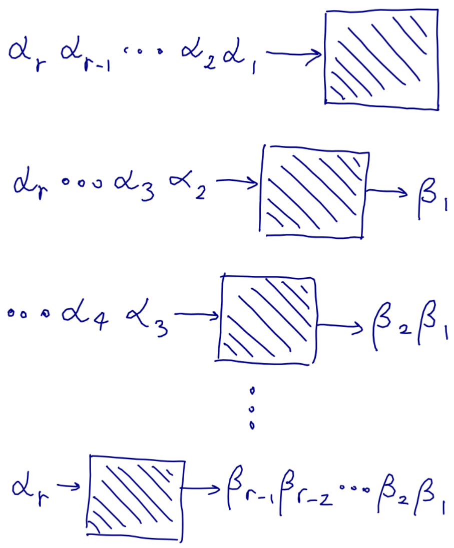
With the notation introduced above, the action of a DMC channel $\mathcal{N}$ on an input RV X gives rise to a pair of dependent RVs (X, Y) with joint probability distribution given by

$$\Pr\{X = x_i \text{ and } Y = y_j\} = p_i p_{ij}.$$

Sometimes we write $Y = \mathcal{N}(X)$.

r-th extension of a DMC: in series

What happens when we feed a string of r symbols $(\alpha_1, \cdots, \alpha_r) \in \Sigma_1^{(r)}$ through a discrete memoryless channel?

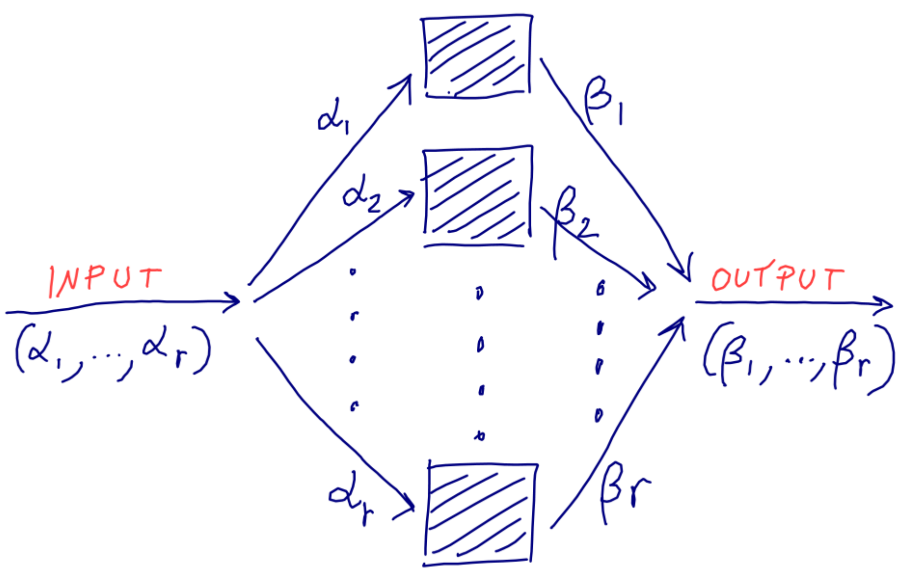


The r-th extension of a DMC is then itself a DMC from an input r-dimensional RV $\boldsymbol{X}$ with range $\Sigma_1^{(r)}$, to an output r-dimensional RV $\boldsymbol{Y}$ with range $\Sigma_2^{(r)}$. The channel matrix is given by the product of the transition probabilities:

$\Pr\{\boldsymbol{Y} = \beta_1 \cdots \beta_r | \boldsymbol{X} = \alpha_1 \cdots \alpha_r\} \stackrel{\text{def}}{=} p(\boldsymbol{Y}|\boldsymbol{X}) = p(\beta_1|\alpha_1) \cdots p(\beta_r|\alpha_r)$.

r-th extension of a DMC: in parallel

We can also think of channel extensions this way:



We have now many copies of the same noisy channel acting 'in parallel.'

Mathematically, serial and parallel extensions are equivalent.

Example: sending a message through a binary symmetric channel

- Imagine that we want to send one word s_i, chosen at random among eight possible words $\{s_1, \cdots, s_8\}$, via a binary symmetric DMC.
- First, we have to encode all words in binary alphabet (the channel only accepts 0s and 1s!).
- $s_1 \mapsto 000, s_2 \mapsto 001, \cdots, s_8 \mapsto 111.$
- Here we use the third extension of the binary symmetric channel (the input consists of three bits.)
- What is the probability that the receiver gets the wrong word? $\Pr\{\text{wrong word}\} = 1 - \Pr\{\text{correct word}\} = 1 - (1-p)^3 = p(3 - 3p + p^2)$. (For $p = 0.5$ is ≈ 0.88; for $p = 0.1$ is ≈ 0.27.)
- Can we do better?

First idea: repetition codes (repeating words)

- Let's try to send each word twice through the channel, i.e., $s_1 \mapsto 000\,000, s_2 \mapsto 001\,001, \cdots, s_8 \mapsto 111\,111.$
- As a decoding rule, if the receiver does not get *the same* word twice in succession, she requests an immediate resending.
- What is the probability of decoding error in this case, i.e., the probability that the receiver gets the wrong word without detecting it?
- First possibility: one error in the first three bits and one error, in the same position, in the second three bits. This contributes with $3 \times p(1-p)^2 \times p(1-p)^2 = 3p^2(1-p)^4$.
- Second possibility: two errors in the first three bits, and two errors, in the same positions, in the second three bits. This contributes with $3 \times p^2(1-p) \times p^2(1-p) = 3p^4(1-p)^2$.
- Third possibility: six errors in a row. This contributes with p^6.
- Total decoding error probability: $p^2(3 - 12p + 21p^2 - 18p^3 + 7p^4)$. (For $p = 0.5$ is ≈ 0.11; for $p = 0.1$ is ≈ 0.02.)
- **But:** it requires feedback from the receiver, for each letter sent.
- **But:** with increasing length, the receiver will *almost always* request a resending.
- **Hence:** zero total decoding error requires infinite repetitions (no reliable communication is possible)

Second idea: parity-check codes

- Instead of just repeating codewords, we can try to exploit another idea.
- **Parity-check coding:** it adds one extra bit (the 'parity bit') at the end of each codeword, so that the sum of the digits is always even.
- In our case, this gives:
  $s_1 \mapsto 000\,0, s_2 \mapsto 001\,1, s_3 \mapsto 010\,1, s_4 \mapsto 011\,0, \cdots s_8 \mapsto 111\,1.$
- If the receiver gets four bits whose sum is odd, she requests an immediate resending. (Hence the name, 'parity-check.')
- What is the probability of decoding error in this case, i.e., the probability that the receiver gets the wrong word without detecting it?
- A wrong decoding happens if there were two or four errors, therefore the decoding error probability is $6p^2(1-p)^2 + p^4$. (For $p = 0.5$ is ≈ 0.44; for $p = 0.1$ is ≈ 0.05.)
- **But:** this code requires feedback.
- **Remark:** this simple idea can be improved, and it is at the basis of some very important families of codes (Low Density Parity-Check, LDPC).

Third idea: Shannon approach (definitions)

- Take a DMC with Σ_1 and Σ_2 as input and output alphabets, respectively.
- An (M, n) code consists of the following:
  1. An index set $\{1, 2, \cdots, M\}$.
  2. An encoding function $\boldsymbol{c} : \{1, 2, \cdots, M\} \to \Sigma_1^{(n)}$ (i.e., each $\boldsymbol{c}_i \stackrel{\text{def}}{=} \boldsymbol{c}(i)$ is a string of n symbols in Σ_1, e.g., $\boldsymbol{c}_i = \alpha_1\alpha_2 \cdots \alpha_n$).
  3. A decoding function $g : \Sigma_2^{(n)} \to \{1, 2, \cdots, M\}$.
- The collection $\mathscr{C} = \{\boldsymbol{c}_1, \cdots, \boldsymbol{c}_M\}$ is called the codebook and its elements are called the codewords. M (the number of codewords) is the size of the code, while n (the length of each codeword) is its length.

Encoding-transmission-decoding: chain of RVs

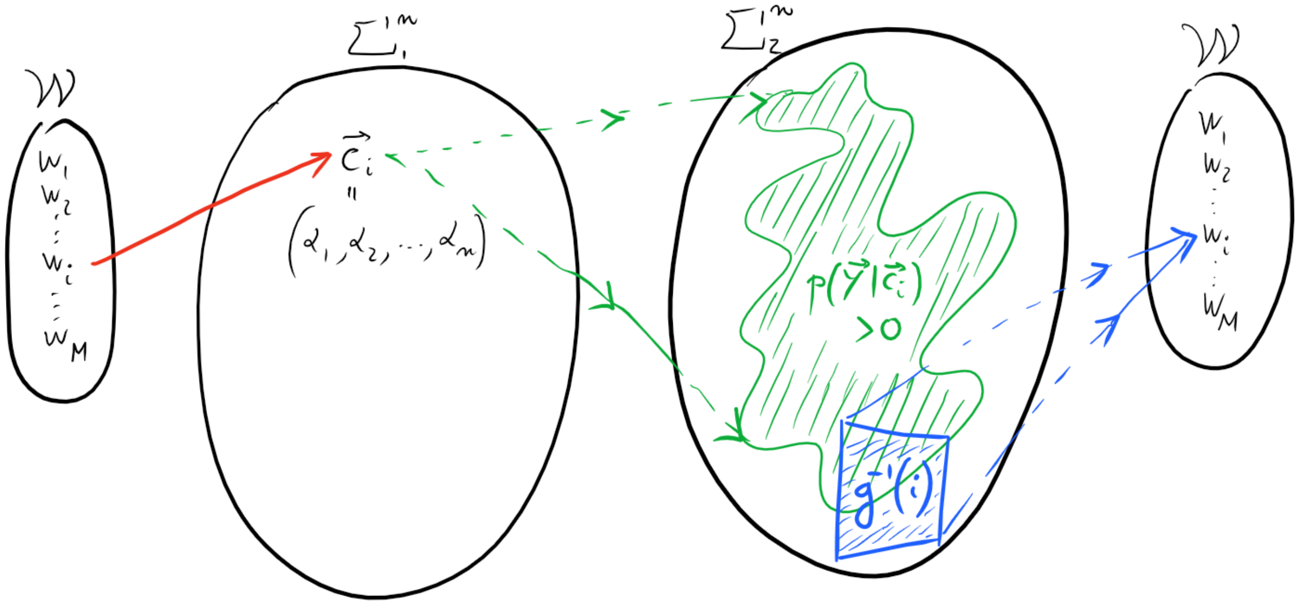
The encoding-transmission-decoding process can be summarized as:

$$W \xrightarrow{\mathcal{E}_n} \boldsymbol{X} \xrightarrow{\mathcal{N}_n} \boldsymbol{Y} \xrightarrow{\mathcal{D}_n} \hat{W}$$

What does this mean?

- W, the message: a RV with range $\{w_1, \cdots, w_M\}$ and probabilities $p_1, \cdots p_M$.
- $\mathcal{E}_n : W \to \boldsymbol{X}$, the length-$n$ encoding: a DMC with input alphabet $\{w_1, \cdots, w_M\}$, output alphabet $\Sigma_1^{(n)}$, and channel matrix given by $p(\boldsymbol{X}|w_i) \overset{\text{def}}{=} \Pr\{\boldsymbol{X} = \alpha_1 \cdots \alpha_n | W = w_i\} = \delta_{\boldsymbol{X}, \boldsymbol{c}_i}$.
- $\mathcal{N}_n : \boldsymbol{X} \to \boldsymbol{Y}$, the n-th extension of the communication channel: a DMC with input alphabet $\Sigma_1^{(n)}$, output alphabet $\Sigma_2^{(n)}$, and channel matrix $p(\boldsymbol{Y}|\boldsymbol{X}) \overset{\text{def}}{=} \Pr\{\boldsymbol{Y} = \beta_1 \cdots \beta_n | \boldsymbol{X} = \alpha_1 \cdots \alpha_n\} = p(\beta_1|\alpha_1) \cdots p(\beta_n|\alpha_n)$.
- $\mathcal{D}_n : \boldsymbol{X} \to \hat{W}$, the decoding: a DMC with input alphabet $\Sigma_2^{(n)}$, output alphabet $\{w_1, \cdots, w_M\}$, and channel matrix given by $p(w_j|\boldsymbol{Y}) \overset{\text{def}}{=} \Pr\{\hat{W} = w_j | \boldsymbol{Y} = \beta_1 \cdots \beta_n\} = \delta_{g(\beta_1 \cdots \beta_n), j}$.
- A decoding error happens whenever $\hat{W} \neq W$. What is the probability that a decoding error occurs?

A picture



encoding noisy channel decoding

Decoding error probability

- How to compute the error probability, i.e. $\Pr\{\hat{W} \neq W\}$?

$$\begin{aligned}
\Pr\{\hat{W} \neq W\} &\stackrel{\text{def}}{=} \sum_{j \neq i} \sum_{i=1}^{M} \Pr\{\hat{W} = w_j, W = w_i\} \\
&= \sum_{i,j=1}^{M} \Pr\{\hat{W} = w_j, W = w_i\} - \sum_{i=1}^{M} \Pr\{\hat{W} = w_i, W = w_i\} \\
&= 1 - \sum_{i=1}^{M} \Pr\{\hat{W} = w_i, W = w_i\} \\
&= 1 - \sum_{i} \sum_{\boldsymbol{X}} \sum_{\boldsymbol{Y}} p(w_i|\boldsymbol{Y}) p(\boldsymbol{Y}|\boldsymbol{X}) p(\boldsymbol{X}|w_i) p_i \\
&= 1 - \sum_{i} \sum_{\boldsymbol{X}} \sum_{\boldsymbol{Y}} \delta_{g(\boldsymbol{Y}),i} p(\boldsymbol{Y}|\boldsymbol{X}) \delta_{\boldsymbol{X},\boldsymbol{c}_i} p_i \\
&= 1 - \sum_{i} \sum_{\boldsymbol{Y} \in g^{-1}(i)} p(\boldsymbol{Y}|\boldsymbol{c}_i) p_i
\end{aligned}$$

- The error probability crucially depends on the choice of the decoding function g.

Ideal-observer (minimum error) decoding

- $\Pr\{\hat{W} \neq W\} = 1 - \sum_j \sum_{\boldsymbol{Y}} \delta_{j,g(\boldsymbol{Y})} p(\boldsymbol{Y}|\boldsymbol{c}_j) p_j$.
- Rewrite $p(\boldsymbol{Y}|\boldsymbol{c}_j)p_j$ as $p(\boldsymbol{c}_j, \boldsymbol{Y}) \stackrel{\text{def}}{=} \Pr\{\boldsymbol{c}_j \text{ sent and } \boldsymbol{Y} \text{ received}\}$.
- Rewrite it again as $p(\boldsymbol{c}_j, \boldsymbol{Y}) = p(\boldsymbol{c}_j|\boldsymbol{Y}) p_{\boldsymbol{Y}}$, where $p_{\boldsymbol{Y}} \stackrel{\text{def}}{=} \Pr\{\boldsymbol{Y} \text{ received}\} = \sum_{j=1}^{M} p(\boldsymbol{c}_j, \boldsymbol{Y})$.
- Then, $\Pr\{\hat{W} \neq W\} = 1 - \sum_{\boldsymbol{Y}} \sum_j \delta_{j,g(\boldsymbol{Y})} p(\boldsymbol{c}_j|\boldsymbol{Y}) p_{\boldsymbol{Y}}$.
- Choose the decoding function $g : \Sigma_2^{(n)} \to \{1, 2, \cdots, M\}$ in such a way that $p(\boldsymbol{c}_{g(\boldsymbol{Y})}|\boldsymbol{Y}) \geqslant p(\boldsymbol{c}_j|\boldsymbol{Y})$, for all $1 \leqslant j \leqslant M$.
- Equivalently: $g(\boldsymbol{Y}) = \arg\max_j p(\boldsymbol{c}_j|\boldsymbol{Y})$.
- This decoding method is called ideal-observer or minimum-error, because it minimizes the error probability.
- **Meaning**: upon receiving $\boldsymbol{Y}$, use this piece of information to infer the most probable codeword.
- The ideal-observer decoding is optimal! **However:** the construction depends on the choice of probabilities $p_1, \cdots, p_M$, which is a serious disadvantage.

Maximum-likelihood decoding

- $\Pr\{\hat{W} \neq W\} = 1 - \sum_j \sum_{\boldsymbol{Y}} \delta_{j,g(\boldsymbol{Y})} p(\boldsymbol{Y}|\boldsymbol{c}_j) p_j$.
- Choose a decoding function $g : \Sigma_2^{(n)} \to \{1, 2, \cdots, M\}$ such that $p(\boldsymbol{Y}|\boldsymbol{c}_{g(\boldsymbol{Y})}) \geqslant p(\boldsymbol{Y}|\boldsymbol{c}_j)$, for all $1 \leqslant j \leqslant M$.
- Equivalently: $g(\boldsymbol{Y}) = \arg\max_j p(\boldsymbol{Y}|\boldsymbol{c}_j)$.
- This decoding method is called maximum-likelihood (ML).
- **Meaning**: upon receiving $\boldsymbol{Y}$, decode it with the codeword $\boldsymbol{c}_i$ that, if sent, maximizes the probability of receiving $\boldsymbol{Y}$.
- Since, in general, $p(\boldsymbol{Y}|\boldsymbol{c}_i) \neq p(\boldsymbol{c}_i|\boldsymbol{Y})$, ML decoding and ideal-observer decoding may give different results.
- **Con**: sub-optimal. **Pro**: independent of the p_i's, much easier to implement.
- **Question**: when do ML and ideal-observer decodings agree?
  Answer: they agree if $p_1 = p_2 = \cdots = p_M = \frac{1}{M}$.

Example: *minimum-error* vs *max-likelihood*

- Suppose you are at the receiver's end of a binary symmetric channel with error probability $\epsilon \stackrel{\text{def}}{=} p_{0\to 1} = p_{1\to 0} = \frac{9}{10}$.
- Suppose you receive a 'zero.' What is the best guess for the input?
- Since the channel introduce an error 90% of the times, one would say: the best guess is that the input was 'one.'
- This is what a max-likelihood strategy says.
- However, imagine that you know that the sender sends 'zero' with probability $p(\text{in=0}) = \frac{19}{20}$ and 'one' with $p(\text{in=1}) = \frac{1}{20}$.
- Then, $p(\text{in=1}|\text{out=0}) = \frac{p(\text{in=1 and out=0})}{p(\text{out=0})} = \frac{\epsilon p(\text{in=1})}{(1-\epsilon)p(\text{in=0})+\epsilon p(\text{in=1})} = \frac{\frac{9}{10}\frac{1}{20}}{\frac{1}{10}\frac{19}{20}+\frac{9}{10}\frac{1}{20}} = \frac{9}{28} \approx 0.32$.
- Therefore $p(\text{in=0}|\text{out=0}) = 1 - p(\text{in=1}|\text{out=0}) \approx 0.68$.
- According to the ideal-observer rule (the optimal one), the best guess is that the input was 'zero.'

Minimum-distance decoding (Hamming distance)

- Let V_n be the set of all binary sequences of length n.
- **Definition**: given $\boldsymbol{x}, \boldsymbol{y} \in \mathsf{V}_n$, their Hamming distance $d(\boldsymbol{x}, \boldsymbol{y})$ is defined as the number of places in which $\boldsymbol{x}$ and $\boldsymbol{y}$ differ.
- **Example**: take V_4 and $\boldsymbol{x} = 0001$ and $\boldsymbol{y} = 1011$; then $d(\boldsymbol{x}, \boldsymbol{y}) = 2$ (first and third digits are different).
- **Minimum-distance decoding**: choose the decoding function $g : \Sigma_2^{(n)} \to \{1, \cdots, M\}$ such that $d(\boldsymbol{Y}, \boldsymbol{c}_{g(\boldsymbol{Y})}) \leqslant d(\boldsymbol{Y}, \boldsymbol{c}_j)$, for all $1 \leqslant j \leqslant M$.
- **Meaning**: upon receiving $\boldsymbol{Y}$, decode it with a codeword $\boldsymbol{c}_i$ that is 'as close as possible' to $\boldsymbol{Y}$, according to the Hamming distance.

Min-Distance ≡ Max-Likelihood (for binary symmetric channels)

Proof. Let $\epsilon \leqslant 1/2$ the bit-flip probability of the channel. For any $\boldsymbol{x}, \boldsymbol{y} \in \mathsf{V}_n$ with $d(\boldsymbol{x}, \boldsymbol{y}) = k$,

$$\Pr\{\boldsymbol{y} \text{ received} | \boldsymbol{x} \text{ sent}\} = \epsilon^k (1 - \epsilon)^{n-k},$$

which is maximum when k is minimum.

Summary of lecture four

- Discrete memoryless channels provide a simple (but very important) model of communication channels
- The coding problem is to design encoding-decoding methods that allow the receiver to guess (with high reliability) the correct input, avoiding errors.
- The optimal decoding method is called ideal-observer decoding, but it is not practical.
- The maximum-likelihood and the minimum-distance decoding are preferable.

Keywords of lecture four

discrete memoryless channel, binary symmetric channel, r-th extension of a DMC, repetition codes, parity-check codes, encoding-transmission-decoding scheme, decoding error probability, ideal-observer decoding, maximum-likelihood decoding, Hamming distance, minimum-distance decoding