Class 26

Pick any collection of k linearly ind vectors from $\mathbb{F}_2^n$ $\{g_1, \ldots, g_k\}$ & put them as rows of a matrix

$$G_{k \times n} = \begin{bmatrix} \underline{g_1} \\ \vdots \\ \underline{g_k} \end{bmatrix}$$

Rowspace (G) = Span (Rows of G) : k-dimensional subspace of $\mathbb{F}_2^n$

↓

valid linear code

'Dimension' of linear code $\mathcal{C}$

$\dim(\mathcal{C})$ $\triangleq$ Dim of subspace $\mathcal{C} \subseteq \mathbb{F}_2^n$ $\leftarrow$

$= k.$

$$|\mathcal{C}| = 2^k.$$

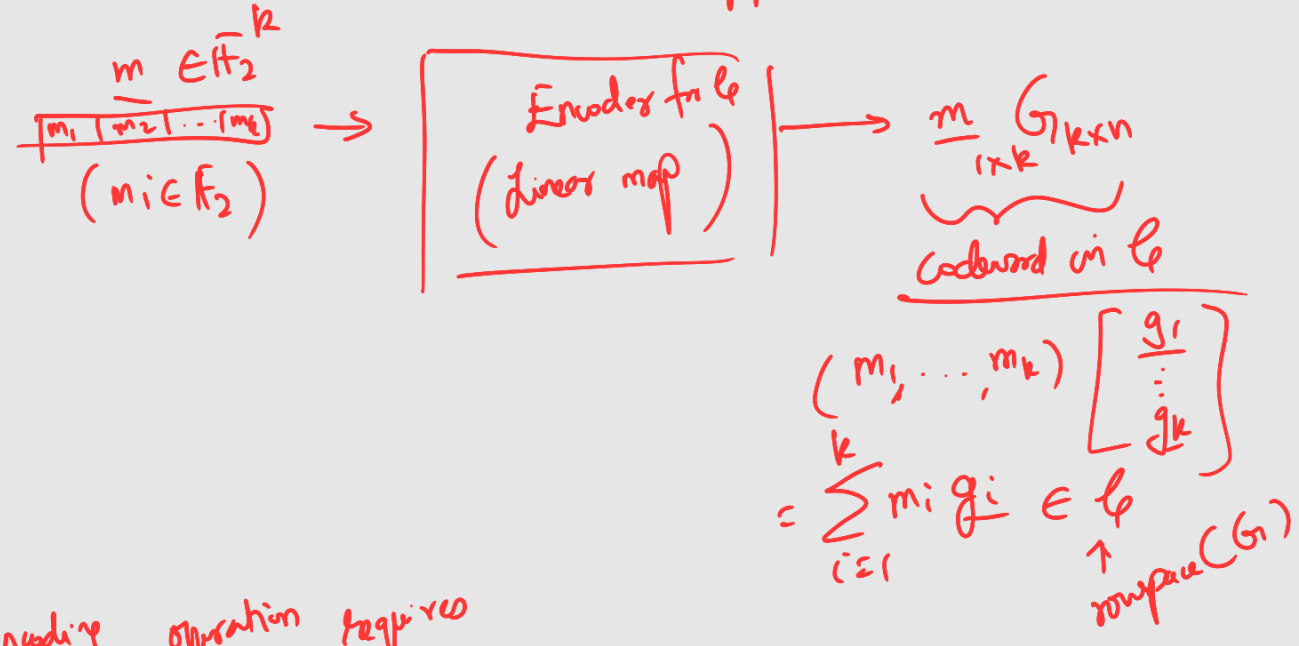
$$R_{\mathcal{C}} = k/n = \frac{\dim(\mathcal{C})}{n}$$

$$d_{min}(\mathcal{C}) = \min_{\substack{\underline{c} \neq 0 \\ \underline{c} \in \mathcal{C}}} w_H(\underline{c})$$

Encoding: → Operation of mapping $2^{nR=k}$ length msgs to the n-length codewords in a unique manner

→ - Mapping from k-length vectors over $\mathbb{F}_2$ to $\mathcal{C} \subseteq \mathbb{F}_2^n$.

→ For linear codes, we can do this encoding as a linear mapping



$$(m_1, \dots, m_k) \begin{bmatrix} g_1 \\ \vdots \\ g_k \end{bmatrix} = \sum_{i=1}^{k} m_i g_i \in \mathcal{C}$$

↑ rowspace(G_1)

→ Encoding operation requires

Storage + Computation polynomial (in n) unlike non-linear codes which in general require exp complexity.

Examples:

Encoding: $m \in \mathbb{F}_2 \rightarrow$ [encoder] $\; mG_1 = (m, \dots, m)$

Repetition code:

$$G_1 = [1\ 1\ 1 \dots 1]_{1\times n}$$

$$\mathcal{C} = \text{Rowspace } G_1 = \{(0 \dots 0), (1, \dots, 1)\}$$

$\dim(\mathcal{C}) = n$, $\dim(\mathcal{C}) = 1 = k$, $R = \frac{k}{n} = \frac{1}{n}$.

Decoding: How to implement min. distance decoder more efficiently?

$$\hat{c} = \underset{c \in \mathcal{C}}{\operatorname{argmin}}\, d_H(y, c)$$

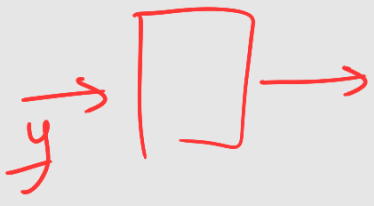
For $n=5$:

Suppose $y = (1\,1\,1\,0\,0)$ → Then min dist decoder (MDD) output is $\hat{c} = (1\,1\,1\,1\,1)$

$$\text{MDD}(y) = \begin{cases} \underline{0} = (0,\ldots,0), & \text{if } w_H(y) < \frac{n}{2} \\ \underline{1} = (1,\ldots,1), & \text{if } w_H(y) > \frac{n}{2} \\ \text{pick } \underline{0} \text{ or } \underline{1} & \text{if } w_H(y) = \frac{n}{2} \end{cases}$$

o/p of MDD given rx vector y as input

Simple Decoding Rule [Majority Logic Decoder]:



Hamming code: This is a class of codes (Binary Hamming Codes)

- $n = 2^q - 1$
- $k = 2^q - 1 - q$
- $d = 3$

for each $q \geq 3$

↓

Pg of linear code

We take up a particular example: (a particular smallest code in the class of Hamming codes)

For $n=3$

$n = 7$, $k = 4$, $d_{min} = 3$.

$G_{\text{Hamming}} = \left(I_4 \;\middle|\; \begin{array}{ccc} 1&0&1 \\ 0&1&1 \\ 1&1&0 \\ 1&1&1 \end{array} \right) \to \left(\text{rowspace}(G_{\text{Hamming}}) \subseteq \mathbb{F}_2^7 \right)$

4×7

↑ appending I_4 with 3 columns to the right.

Note: The 4 rows of G are linearly independent vectors of $\mathbb{F}_2^7$ ($\text{Rank}(G)$ = no of linearly ind vectors in rows (or cols) = no of lin ind rows $= 4$)

$\mathcal{C} = \text{rowspace}(G)$ is a 4-dim linear code.

Rate $= 4/7$, $|\mathcal{C}| = 2^k = 2^4 = 16$

$d_{\min}(\mathcal{C})$ = min wt of non-zero codewords $= 3$ (please verify)

Next class: Decoding of Hamming Codes. (this code can correct single errors $t = 1 = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor$)

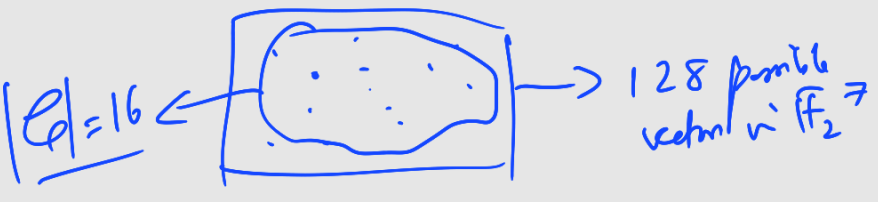
Class 27

$\underset{1\times 4}{m}$ = message vector = (0 1 0 1) → all possible vectors in $\mathbb{F}_2^4$. $\longrightarrow$ mapped to codeword (0 1 0 1) G

$= 0 \cdot g_1 + 1 \cdot g_2 + 0 \cdot g_3 + 1 \cdot g_4$

(g_i = i^{th} row of G)

= (0, 1, 0, 1, 1, 0, 0) → codeword corresponding to m = (0 1 0 1).

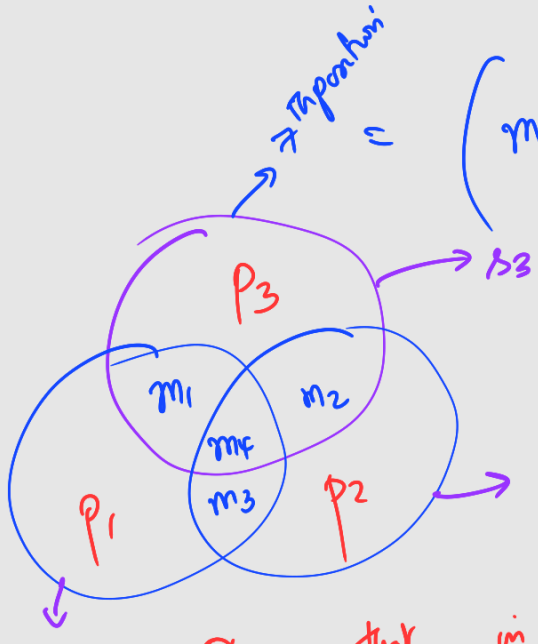


For arbitrary message $\underline{m} \in \mathbb{F}_2^4$

(m_1, m_2, m_3, m_4)

$$\text{Codeword} = (m_1, \ldots, m_4) \left(I_4 \;\middle|\; \begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{matrix} \right)$$

$$= (m_1, m_2, m_3, m_4, \underbrace{m_1+m_3+m_4}_{p_1}, \underbrace{m_2+m_3+m_4}_{p_2}, \underbrace{m_1+m_2+m_4}_{p_3})$$



redundant bits or parity bits.

Observe that in the codeword $= (c_1, c_2, c_3, c_4, c_5, c_6, c_7)$

$= (m_1, \ldots, m_4, p_1, p_2, p_3)$

parity equations

$$\begin{cases} c_5 + c_1 + c_4 + c_3 = 0 \\ c_7 + c_1 + c_4 + c_2 = 0 \\ c_6 + c_2 + c_3 + c_4 = 0 \end{cases} \rightarrow \text{For all valid codewords.}$$

Guarantee is to correct 1 error:

Such a codeword $\underline{c}$ is sent through the channel (assume that no of errors ≤ 1). Let Rx vector be $\underline{y} = (y_1, \ldots, y_7) \in \mathbb{F}_2^7$.

Note $d_H(\underline{c}, \underline{y}) \leq 1$

We already know how to implement MDD to recover $\underline{c}$ from $\underline{y}$.

↳ 16 comparisons for MDD

Is there a simpler decoder?

Hamming code decoder:

Decoder calculates $s_1 = y_5 + y_1 + y_4 + y_3$ & $s_3 = y_6 + y_2 + y_3 + y_4$

$s_2 = y_7 + y_1 + y_4 + y_2$

(s_1, s_2, s_3) → Syndrome vector → This decoding technique is "Syndrome Decoding"

Suppose no errors happened $d_H(y, c) = 0$,

Then $(s_1, s_2, s_3) = (0, 0, 0)$

We will show that each position of the error (of the 7 positions) will correspond to 7 possible non-zero vectors for (s_1, s_2, s_3) → Please check.

Thus decoder can find out which position is in error by computing the (s_1, s_2, s_3) vector.

→ This is simpler than 'brute-force' MDD.

→ 'Brute force' MDD is not utilizing the structure of the code, whereas the Hamming Code Decoding uses the linear code structure

→ Computing the syndrome will give the error position

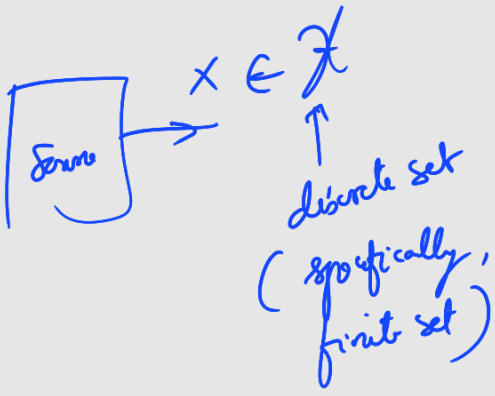
→ Flip that position in y to give the correct transmitted codeword c.

→ Decoded codeword for c. To decode the msg vector m, take the first 4 bits of c. (as right inverse of G_T).

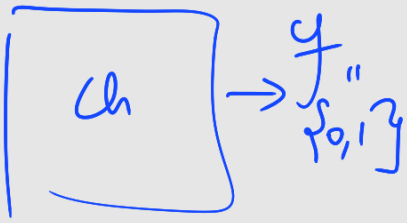
—x—

Till now we have focussed on

(a) Discrete sources : (More specifically, finite alphabet sources)



(b) Discrete Channel (specifically binary channel) $\mathcal{X} = \{0,1\}$

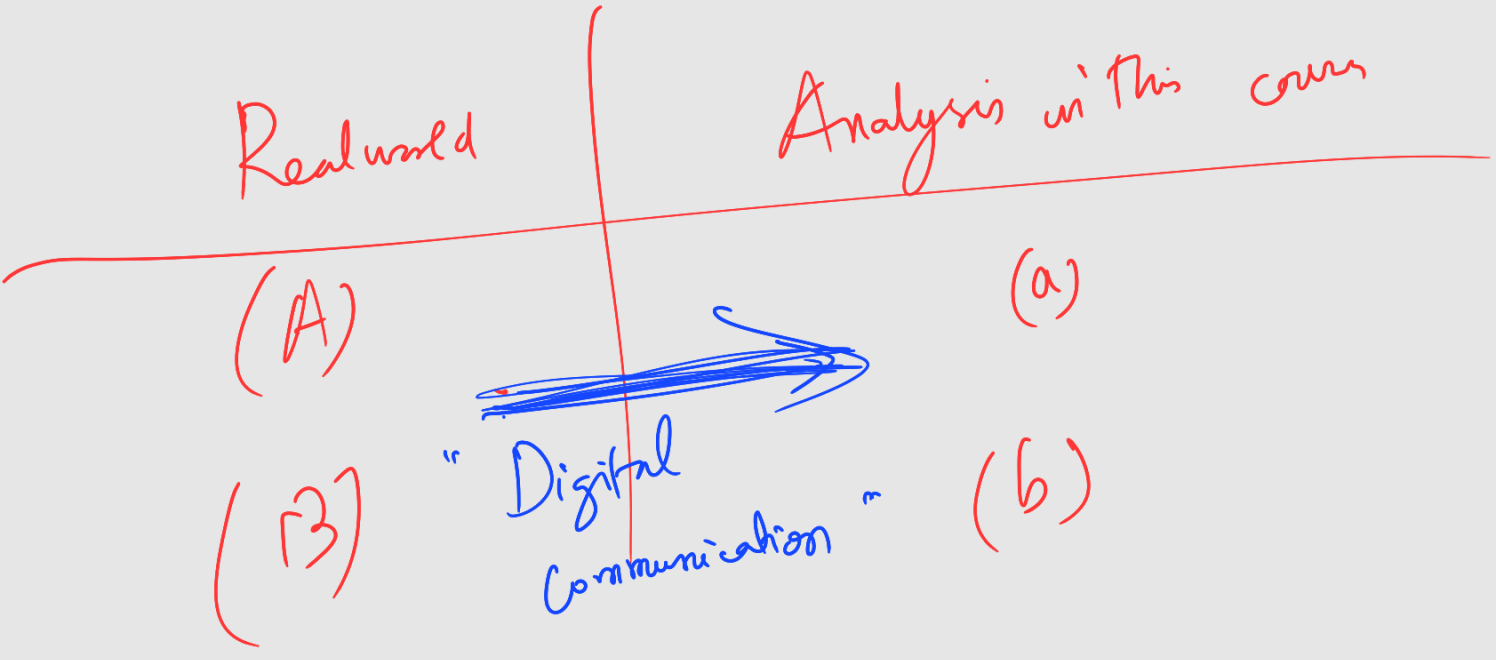


However

(A) Many real-world sources generate analog signals in practice

(B) Real-world channels carry analog signals (electric signals, EMW)

So what is connection b/w (A) and (a) & (B) and (b).



By building the engineering system using digital comm principles, we can translate (A) to (a) and (B) to (b) } for analysis of performance very easily

Natural Resources $ ≤ Money Cost Effort

reduction!

(A) (B) Synthesis ← Engineering implementation

This analysis helps us in figuring out what works in a engineering design

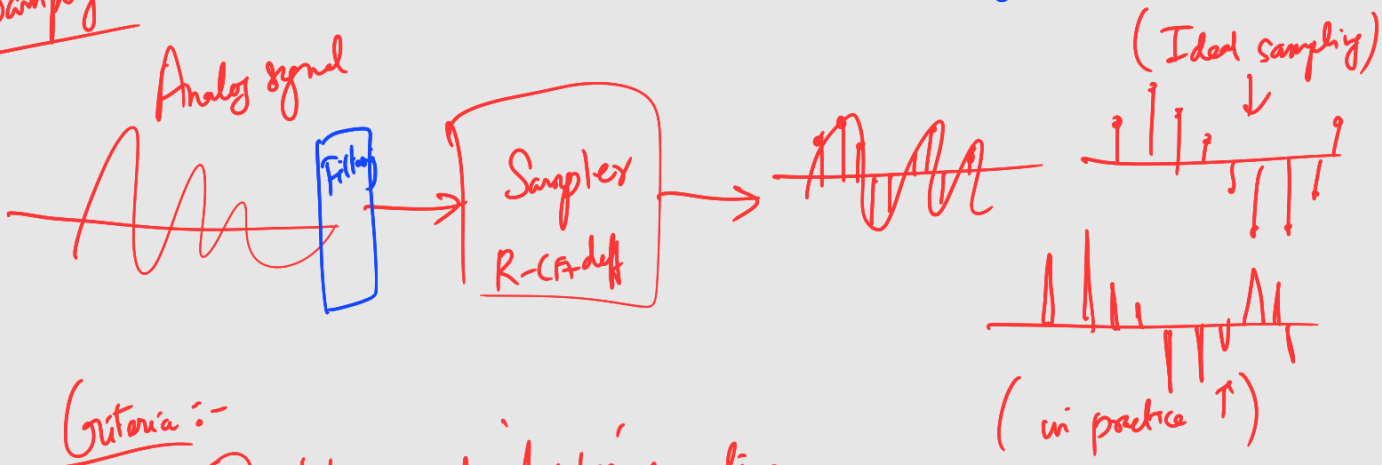
Class 28

Conversion of analog signal sources to discrete sources (specifically binary sources).

Analog signal → [Sampling (to convert signal into discrete time) + Quantize (to convert samples into discretized sample (finite alphabet $\mathcal{X}$))] → Signal that is discrete in time, & at each time instant it takes one value among in finite alphabet $\mathcal{X}$.

(electrical signal. If not electrical, we use a transducer to convert the source signal with an equivalent analog signal)

Sampling:



Criteria :-

① We want lossless sampling

↓

From the samples, we should be able to reconstruct the signal from the sampled signals.

② Suppose signal bandwidth = B Hz

[$B \triangleq$ Max freq (useful) with non-zero content − Min freq (useful) with non-zero content]

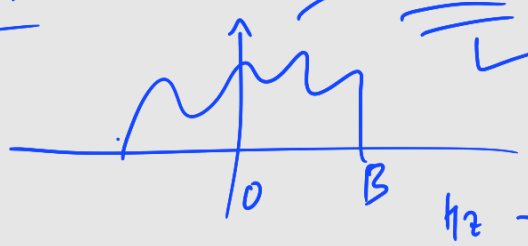
"useful": spurious frequency components filtered before sampling

⇐ [For lossless sampling, we need the sampling rate $\left(\frac{1}{\text{Sampling period}}\right)$ to be $> 2B$]

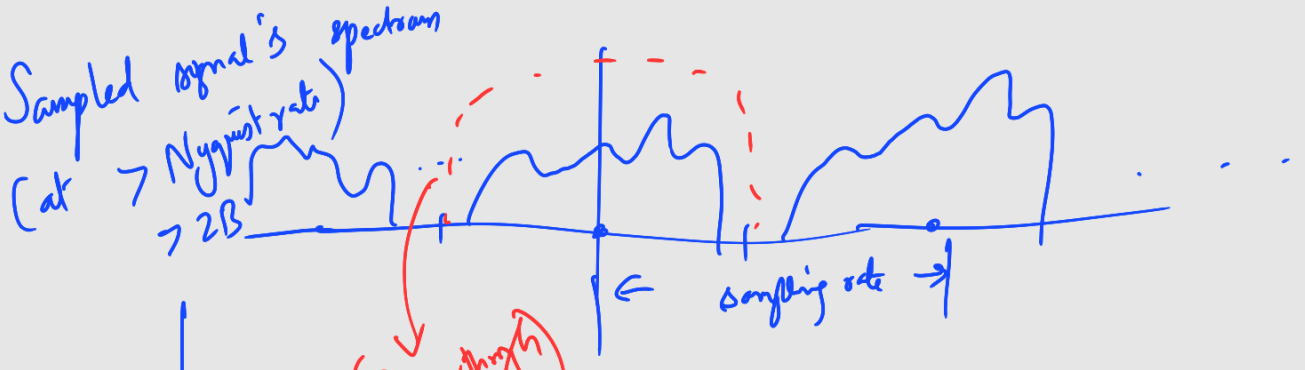
⇐ Nyquist Sampling Theorem

(Royth)

Proof of Sampling theorem:



→ 'baseband' signal
→ signal frequencies from 0 – B Hz

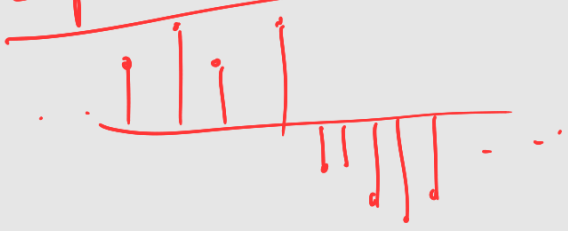


Passing Sampled signal (Recovering through LPF) through a LPF of bandwidth $(2B+\epsilon)$ $(B+\epsilon$ on-fve side$)$

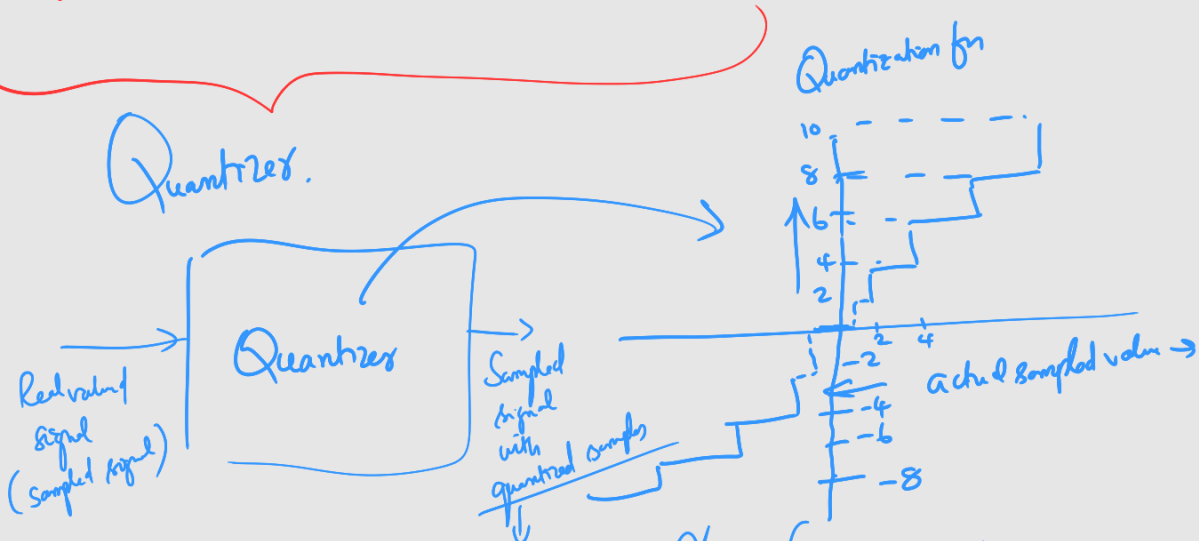
we get the original signal

// Sampling is done

Sampled signal (sample values $\in \mathbb{R}$) → We want to restrict the values to a finite (discrete) set.



Quantizer.



Quantz. samples $\in \mathcal{X}$.

↳ For this ex:

$\mathcal{X} = \{-8, -6, -4, -2, 0, 2, 4, 6, 8, 10\}$

// Quantization (we have a discrete signal now)

Quantized signal $X \to$ [Source coding] $\to$ binary codewords

$\mathcal{X}$

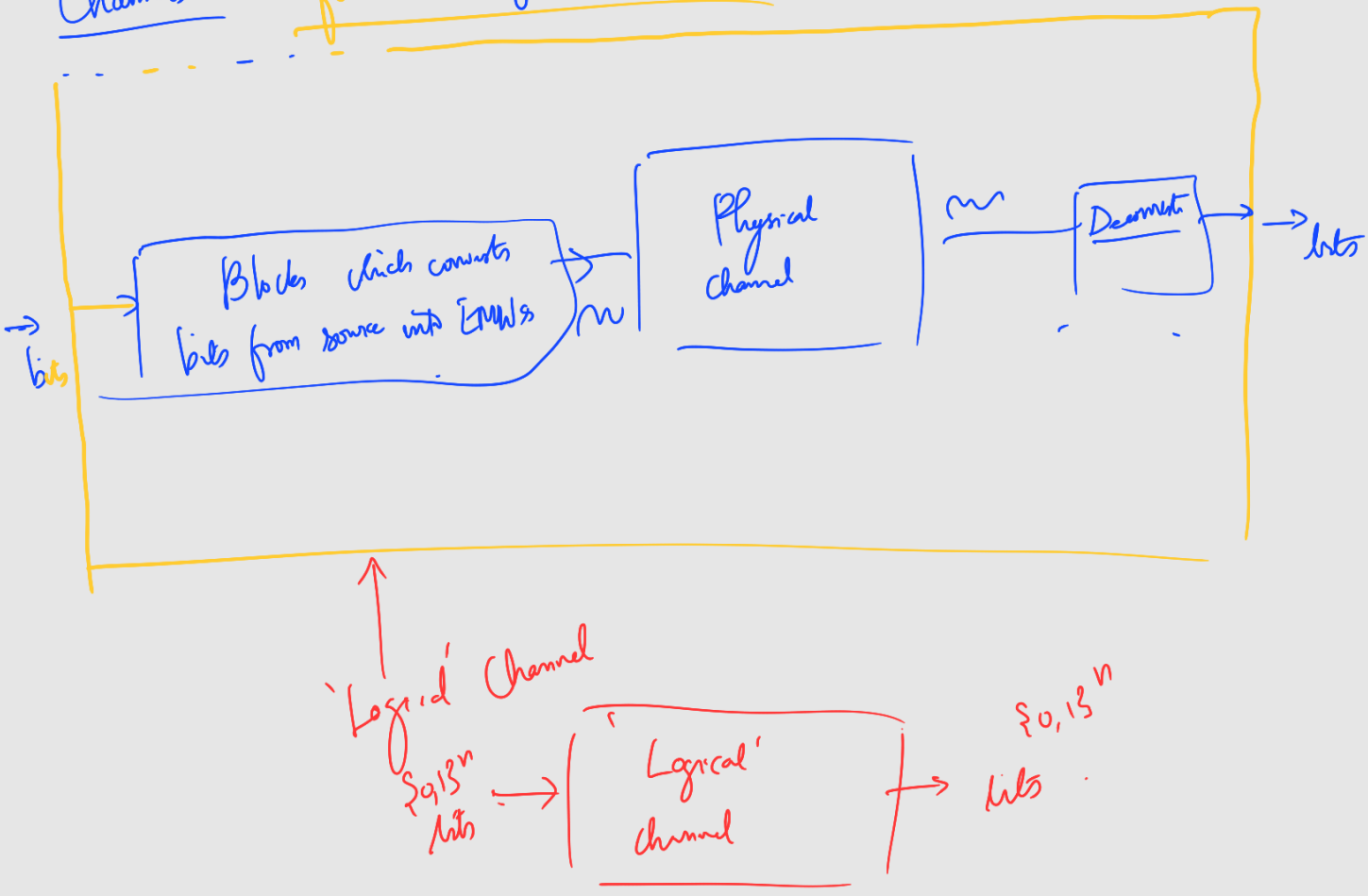
Source coding: ① If we know / can calculate P_X, then we can do source coding 'intelligently' (like prefix-free codes)

By using the statistics of the signal, we can approximate P_X.

(Prog assignment has an example)

② If we don't know P_X, simply represent each $x \in \mathcal{X}$ using a unique binary string of length $\lceil \log_2 |\mathcal{X}| \rceil$

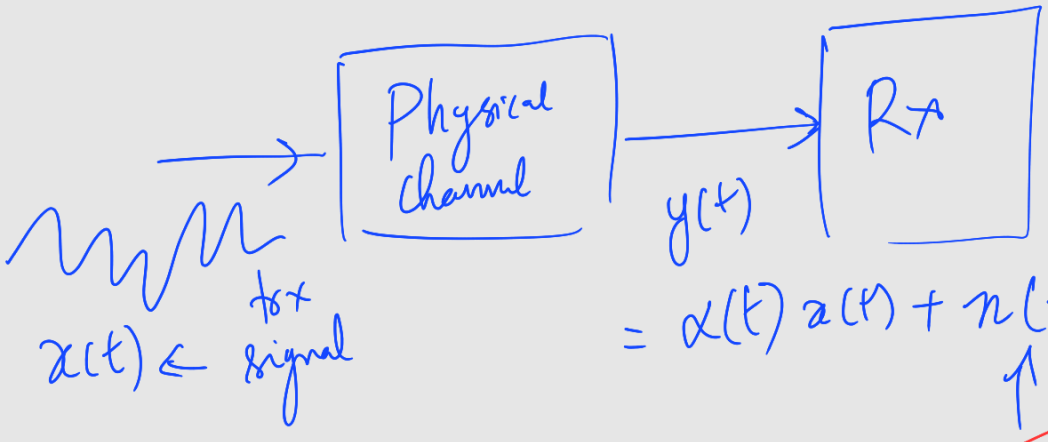
Channels: from analog to discrete



'Logical' Channel

$\{0,1\}^n$ bits $\to$ ['Logical' channel] $\to$ $\{0,1\}^n$ bits

Class 29 :



Some statistics are known about this

$= \alpha(t)\, x(t) + n(t)$

Random process

noise signal

Wireline communication

$\alpha(t)$ → Suppose it is a const

$\alpha(t) = \alpha, \forall t$, we can handle it by boosting signal power.

So we simplify ch. model as $y(t) = x(t) + n(t)$

Over discussion

$\alpha(t)$ is a fn of time

↳ This corresponds to Wireless Channel model

⇒ This also means $\alpha(t)$ is another Random Process

↓

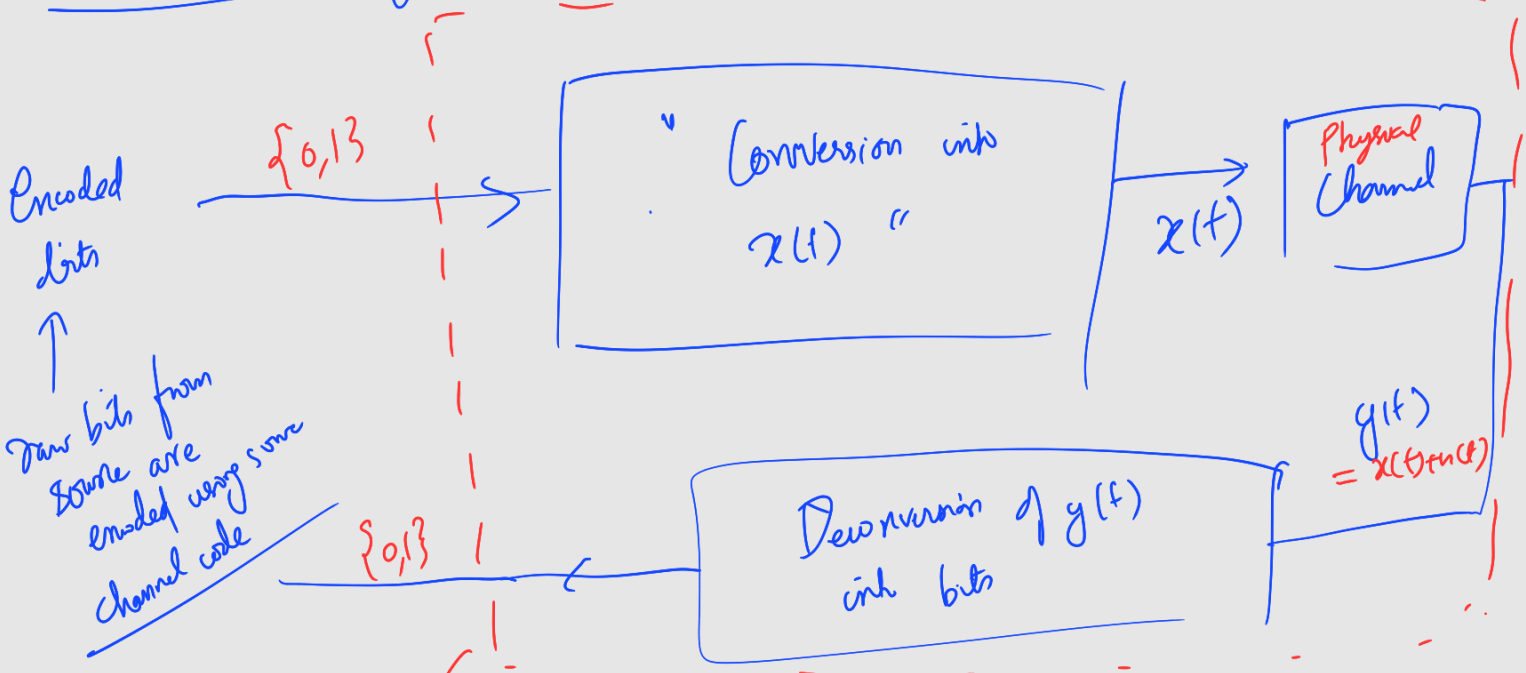
Rx knows some statistics but not actual signal.

Take Wireless Comm course

$x(t)$ is the actual signal Rx is interested in.

How is $x(t)$ generated ?

BSC



'Binary' aspect is clear. How is it Symmetric?

↓

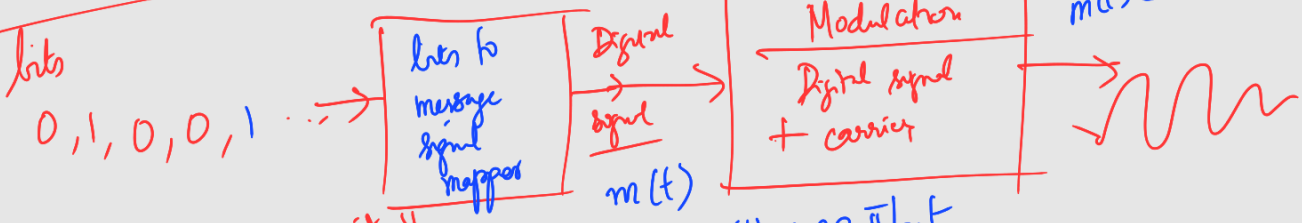
It becomes symmetric under some assumptions on (a) the physical channel

(b) how we do the conversion deconversion

↓

Conversion / Deconversion process of bits → tx signal ?
Rx signal → bits

Conversion of bits → signals :



k^{th} symbol" Energy per bit "

$0 \rightarrow -a = a^2$

$1 \rightarrow a = a^2$

message signal

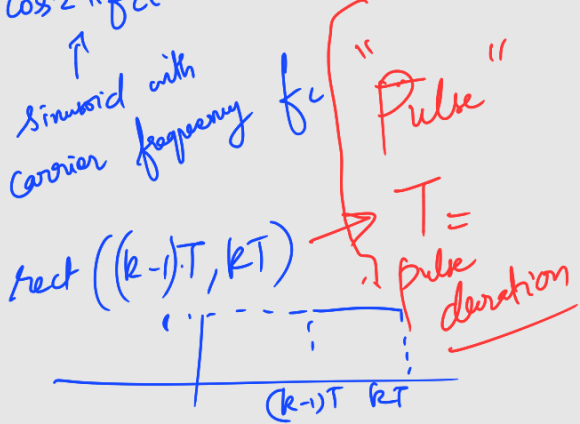
$\text{rect}((k-1)T, kT)$

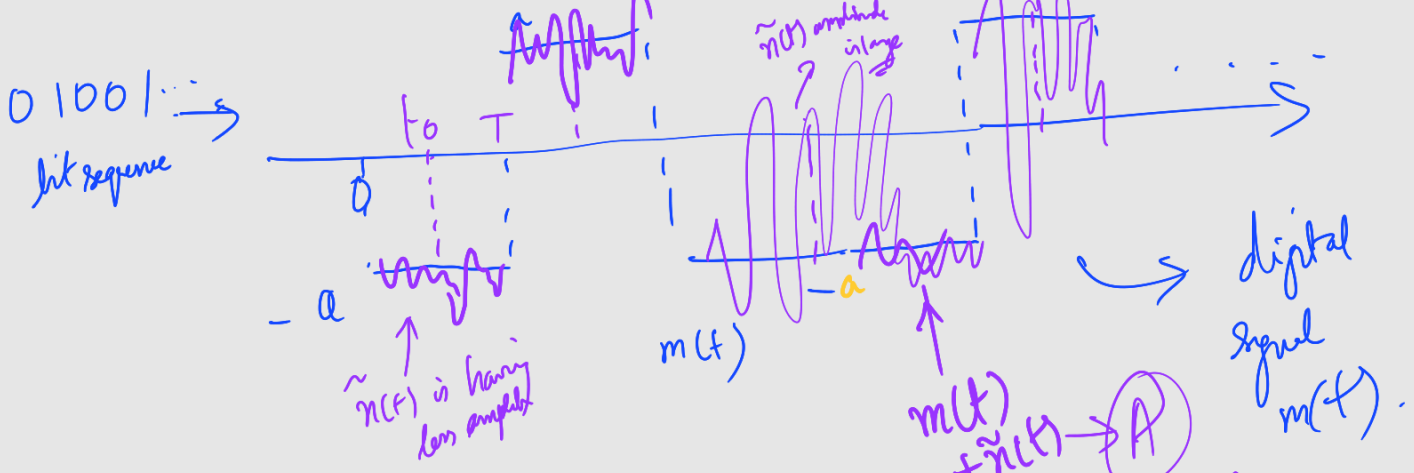
$\text{rect}((k-1)T, kT)$

$c(t) = \cos 2\pi f_c t$

↑

Sinusoid with carrier frequency f_c





Tx signal $x(t) = m(t)\, c(t)$

$= m(t) \cos 2\pi f_c t$

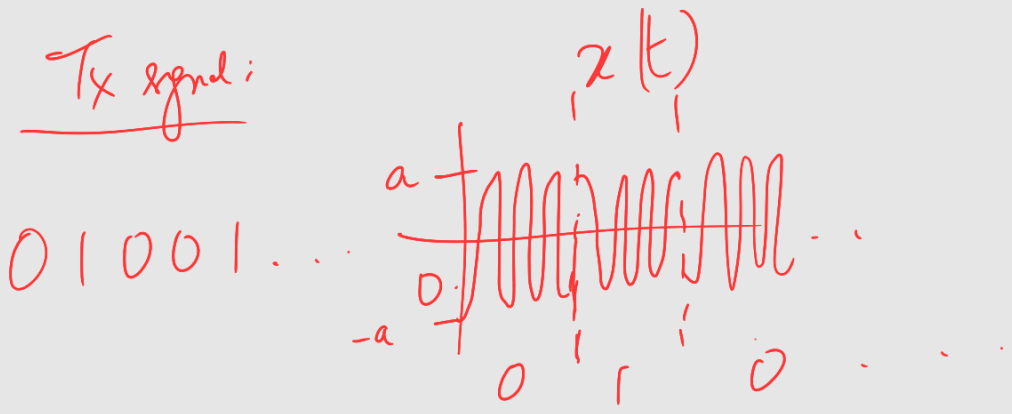
Tx signal

k^{th}

$0 \longrightarrow -a \cos 2\pi f_c t$ $\quad t \in [(k-1)T, kT]$

$1 \longrightarrow +a \cos 2\pi f_c t$ $\quad$ "

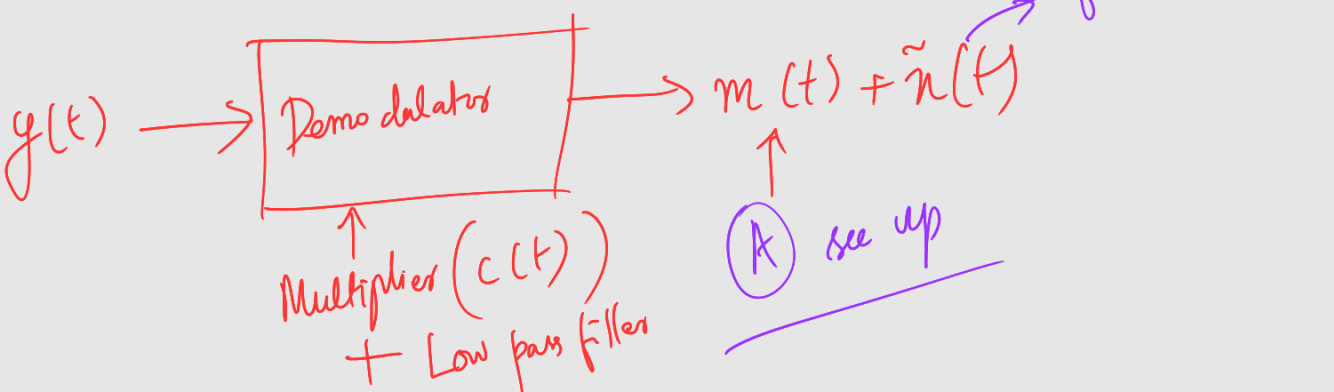
Tx signal:

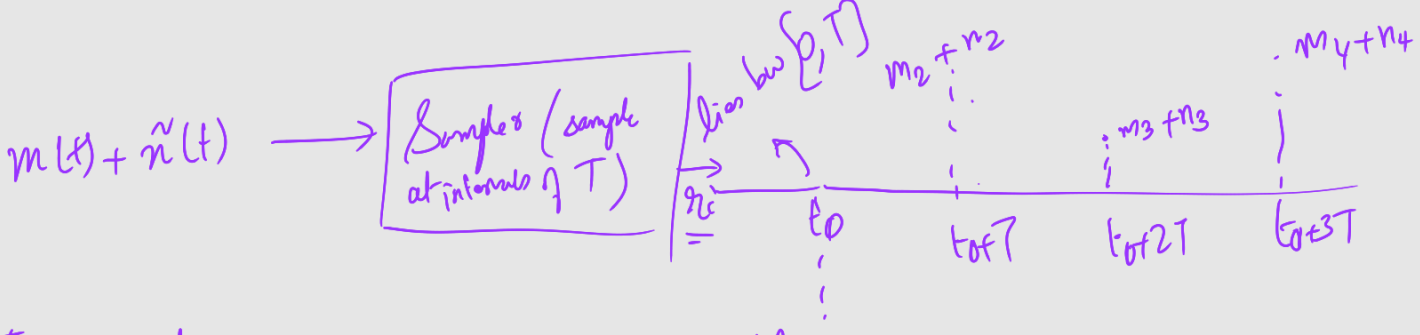


Received signal

$m(t)\cos 2\pi f_c t$

$y(t) = x(t) + n(t)$





Tx samples

$m_i \in \{+a, -a\}$

m_i = samples of $m(t)$ (pure message) at $t_0+(i-1)T$

Rx noisy samples are

$$r_i = m_i + n_i = \begin{cases} a+n_i \\ -a+n_i \end{cases}$$

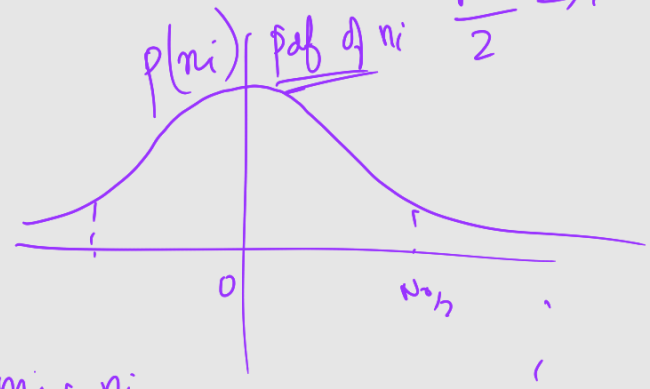
n_i = samples of $\tilde{n}(t)$ (noise signal) at $t_0+(i-1)T$

n_i is the sample of a R. Process → will be a R Variable

We will assume that $n_i \sim$ Gaussian distributed

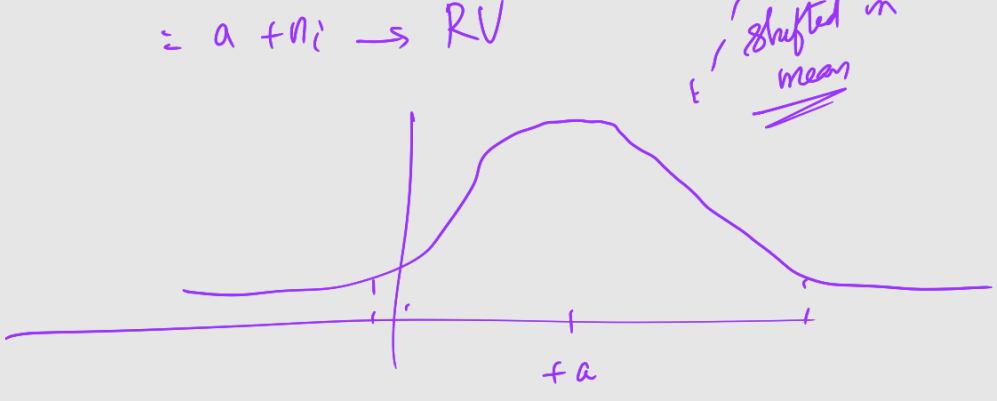
$$n_i \sim N(\mu, \sigma)$$

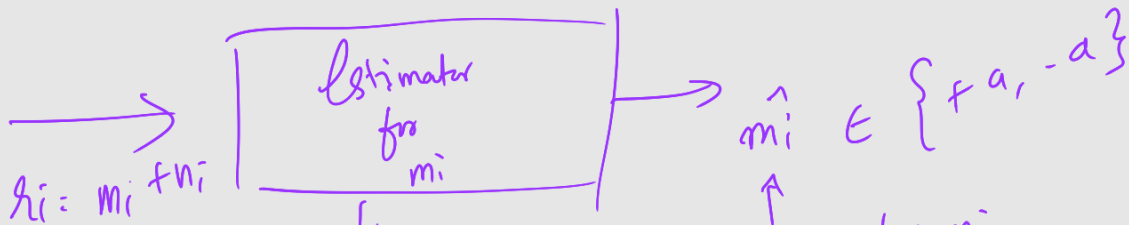
$\mu = 0$, $\sigma = \frac{N_0}{2}$ → some constant



If $m_i = +a$, then $r_i = m_i + n_i$

$= a + n_i$ → RV





ML estimate
Max likelihood

estimate for m_i

$$\hat{m}_i = \underset{m \in \{a,-a\}}{\operatorname{argmax}} \frac{P(r_i \mid Tx\ was = m)}{\text{Prob of getting } r_i \mid Tx = m}$$

$r_i = m_i + n_i$

$$P(R_x \text{ value} = r_i \mid Tx = a) = P(\text{noise} = r_i - a)$$

$$P(R_x = r_i \mid Tx = m) = P(\text{noise} = r_i - m)$$

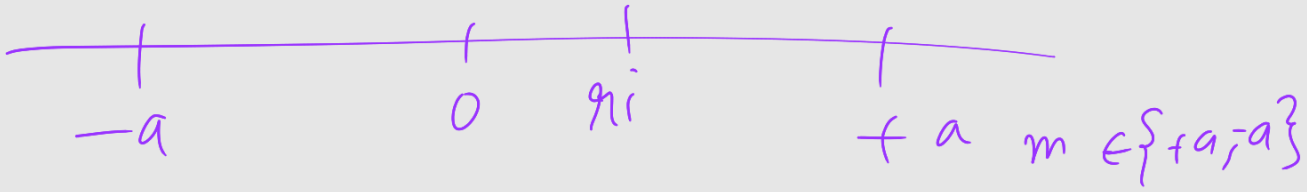
$$P(R_x \text{ value} = r_i \mid Tx = -a) = P(\text{noise} = r_i + a)$$

Plugin the noise pdf (gaussian)

$$\underset{m \in \{+a,-a\}}{\operatorname{argmax}} P(R_x = r_i \mid Tx = m)$$

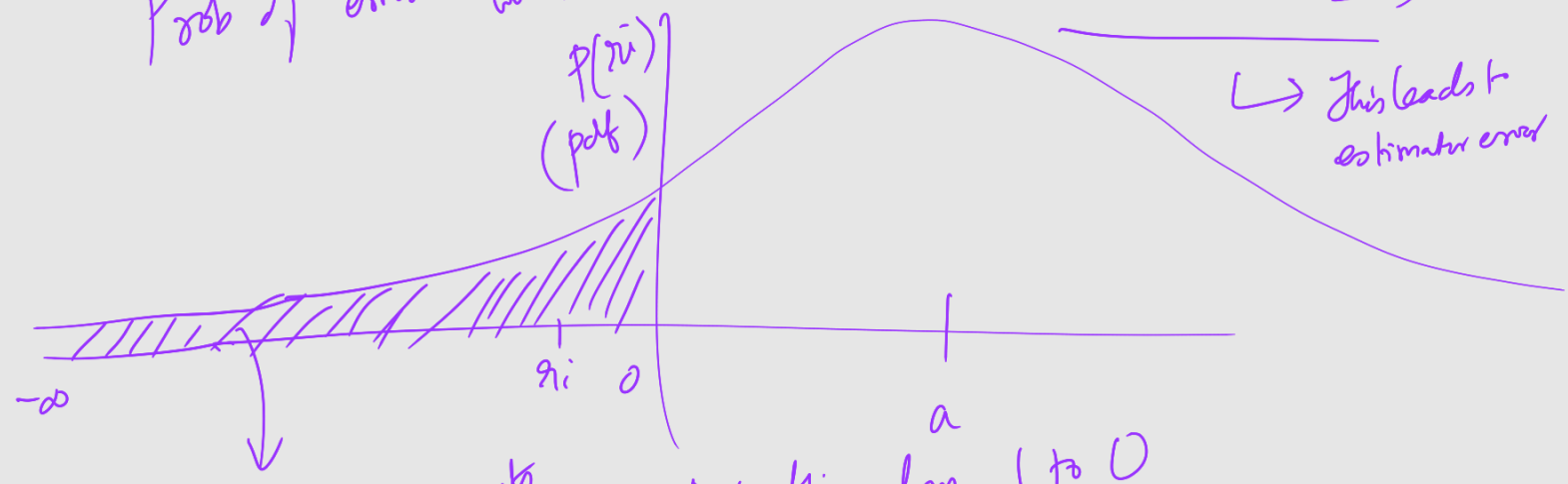
$$= \underset{m \in \{+a,-a\}}{\operatorname{argmin}} |(r_i - m)|$$

→ ML decodes outputs either $\{-a, a\}$ whichever closer

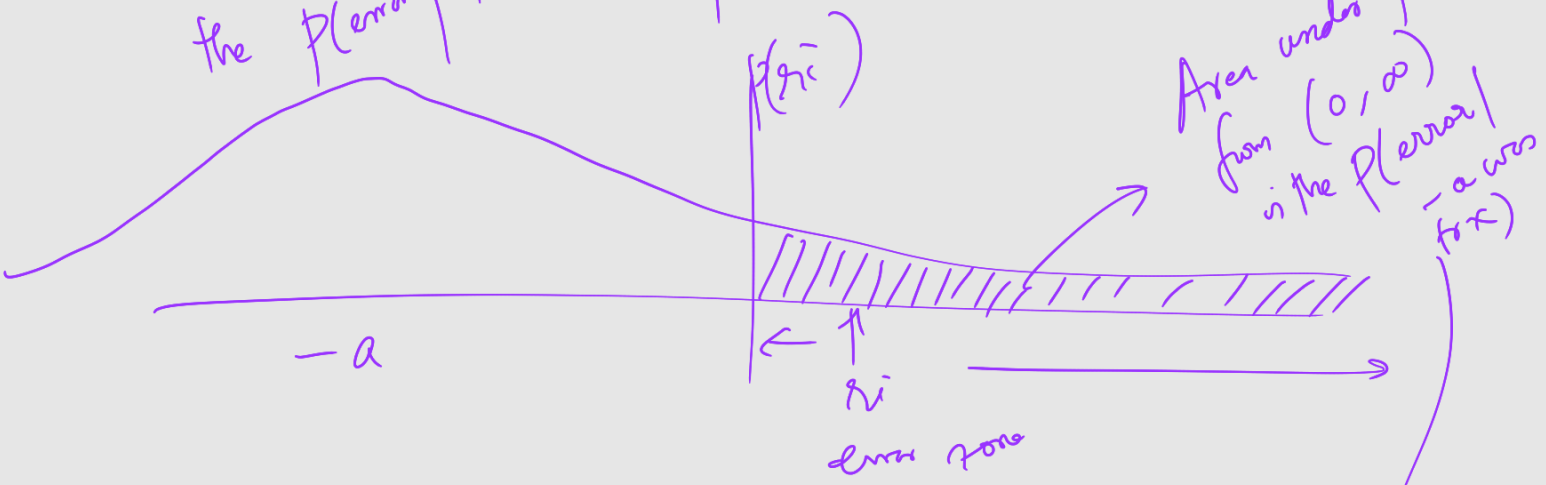


Prob of error when $Tx = a = P(y_i \in (-\infty, 0])$

↳ This leads to estimator error



This area represents the $P(\text{error} \mid Tx = a)$ = bit flip prob from 1 to 0

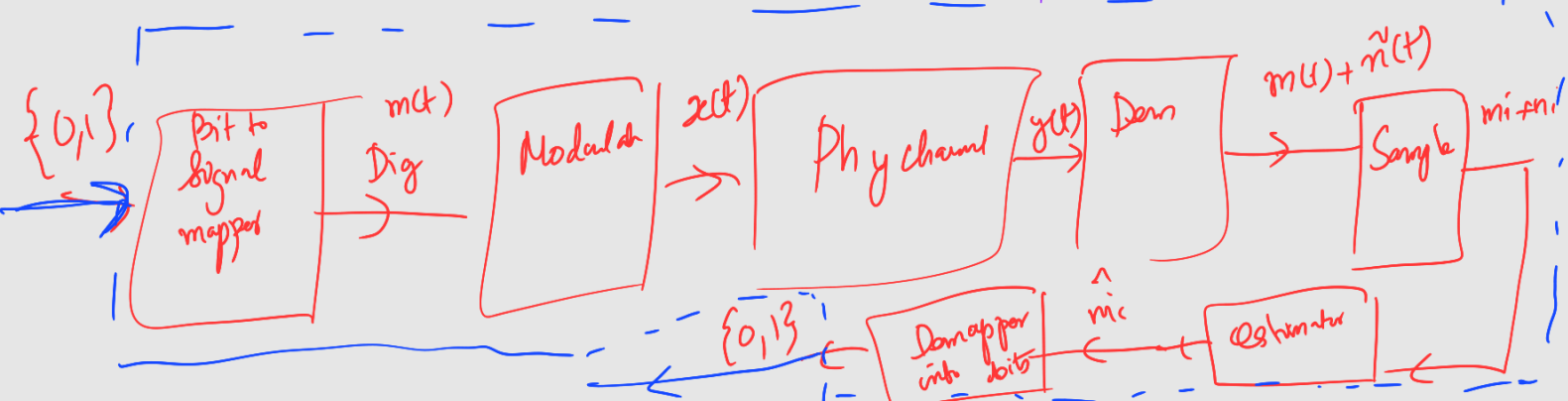


bit flip prob from 0 to 1

Both the areas are same

$\Rightarrow$ Bit flip prob from $0 \to 1$ & $1 \to 0$ are exactly same

Fix this value as p




{0,1}
BSC
{0,1}
with bit flip prob = p