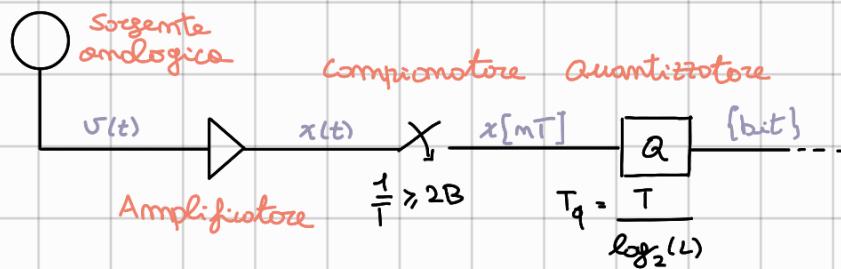


Terzo dei codici

Andiamo avanti con lo schema del trasmettitore:



Il prossimo blocco da analizzare sarebbe il **codificatore di sorgente** (che però soltanto), componente con il compito di comprimere il segnale, riducendo la quantità di bit da trasmettere.

introduce ridondanza

Il blocco successivo, che andiamo invece ad approfondire, è il **codificatore di parole**, questo componente ha il compito di codificare i bit ricevuti in ingresso utilizzando un certo **codice**, con l'obiettivo di aumentare la resistenza del segnale a rumore ed interferenza, favorendo la **rivelazione e la correzione di errore**.

In particolare tratteremo **codici lineari** (utilizzano operazioni tra matrici) e **blocki** (lavorano su parole di bit), caratterizzati dal **rate** $R = \frac{K}{m}$:

- K = dimensione in bit delle parole in ingresso
- m = dimensione in bit delle parole in uscita

chiaramente visto che dobbiamo introdurre ridondanza $m > K$

Esempio

Qual è il rate di un codice che ha 1024 parole di lunghezza 15 bit?

Vuol dire che il codice ha come possibile output 1024 parole diverse

1 bit = 1 bit / 15 bit = 1/15 = 0.06666666666666666

codificare su 15 bit, supponendo $m=75$, con che possiamo codice binari
dove esserci una corrispondenza biunivoca tra le parole di ingresso
e di uscita, supponendo deve avere esattamente 1024 possibili ingressi
e questi si riescano a codificare su $k=10$ bit, per cui $R = \frac{10}{15}$

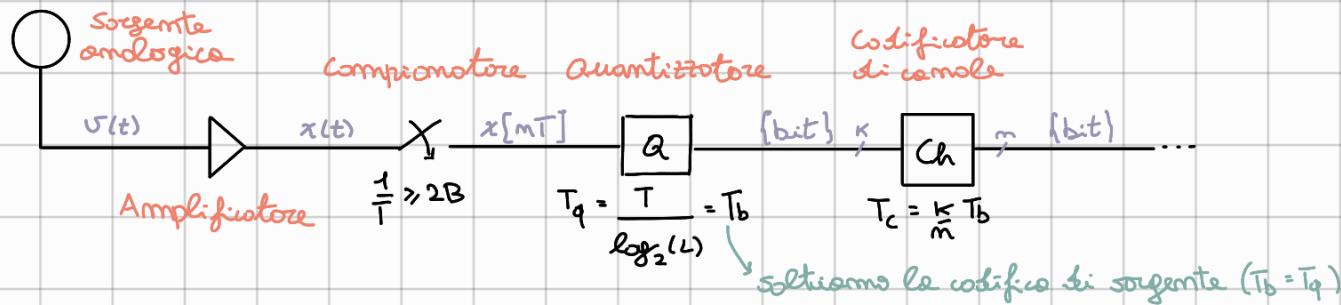
Note: Il canale è il mezzo fisico attraverso il quale si trasmette il segnale
al ricevitore, in particolare noi utilizzeremo sempre canali:

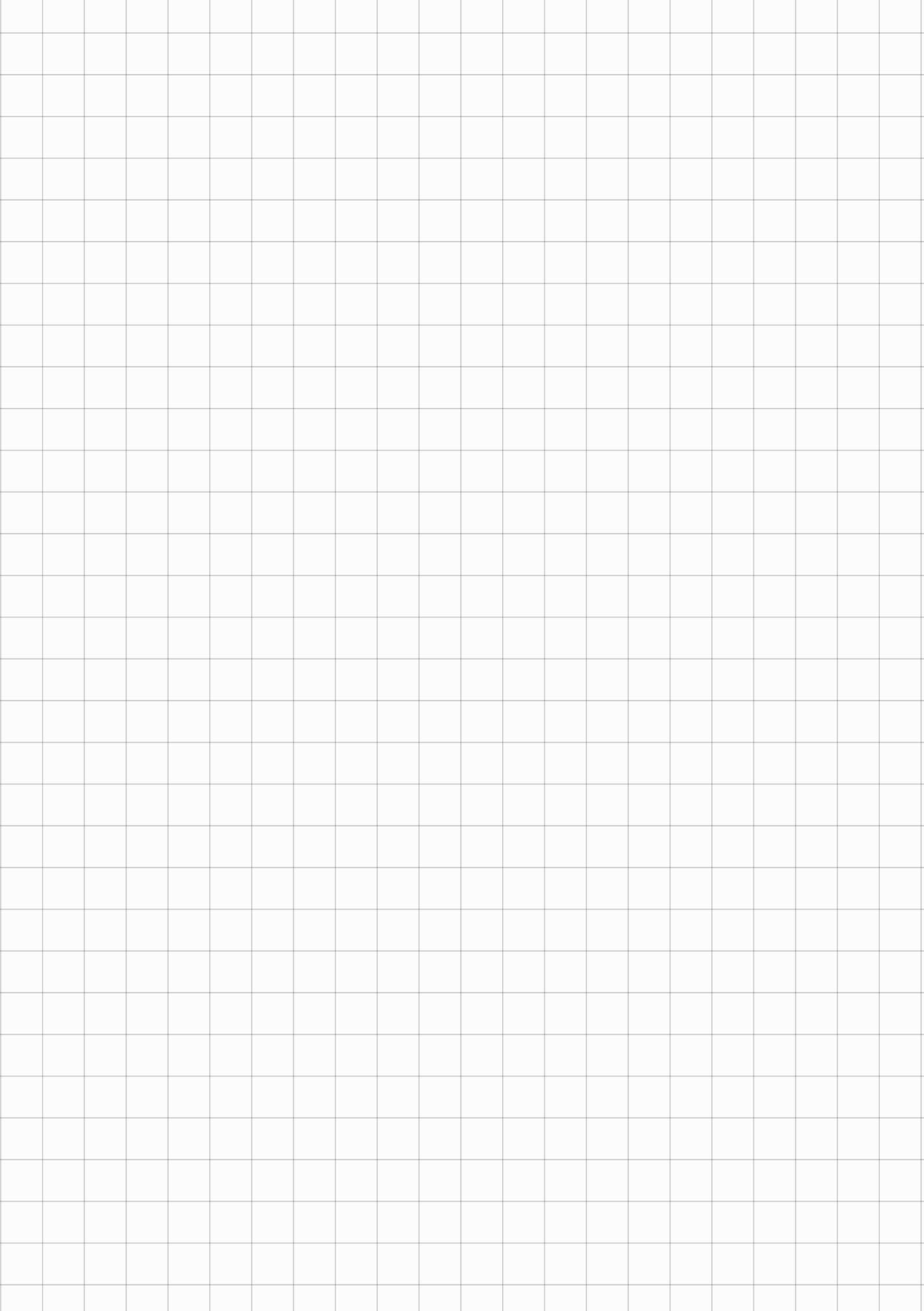
→ **binari**: trasmettiamo bit

→ **simmetrici**: le probabilità di errore sono uguali per D e f

→ **senza memoria**: ogni bit viene trasmesso con una probabilità di
errore indipendentemente dagli altri

Inserendo i componenti citati il trasmettitore diventa quindi:





Codici lineari a blocchi

Innanzitutto questi codici utilizzano l'aritmetica modulo 2, ed è bene ricordare che in tale aritmetica l'addizione è uguale alla sottrazione, infatti da $1+1=1_{(2)}=0 \Rightarrow 1=-1$

Sia $m = [m_1, \dots, m_k]$ una parola di k cifre binarie, il codice a blocco lineare $C(k, n)$ è l'insieme delle 2^k parole $c = [c_1, \dots, c_m]$ di m cifre binarie ottenute con la trasformazione lineare $c = mG$ dove G è detta matrice generatrice del codice, in altre parole $c = mG = \sum_{i=1}^k m_i g_i$, di conseguenza:

- Ogni parola di codice è una combinazione lineare delle righe della matrice generatrice (per avere 2^k diverse G deve avere rango k) righe di G sono base di C
- Il codice è costituito da tutte le possibili combinazioni delle righe della matrice generatrice
- La somma di due parole di codice è ancora una parola di codice
i-esima di 2^k
Dim: Siamo $c_i = m_i G$ e $c_j = m_j G$, allora $c_i + c_j = m_i G + m_j G = m_k G = c_k$
- La m-upla di tutti zeri è sempre una parola di codice
Dim: $\vec{0}$ è sempre m_0 ed $m_0 G = \vec{0}$

La distanza di Hamming $d_H(c_1, c_2)$ fra due vettori di m elementi c_1 e c_2 è definita come il numero di posizioni in cui le due parole sono diverse tra loro, questa metrica gode delle seguenti proprietà:

- $d_H(c_1, c_2) \geq 0$
- $d_H(c_1, c_2) = 0 \Leftrightarrow c_1 = c_2$
- $d_H(c_1, c_2) = d_H(c_2, c_1)$
- $d_H(c_1, c_3) \leq d_H(c_1, c_2) + d_H(c_2, c_3)$

Si definisce inoltre peso di Hamming $w(c) = d_H(c, \vec{0}_m)$

La distanza minima di un codice C è la minima distanza di Hamming fra tutte le possibili parole di codice, in particolare questo equivale al minimo peso di Hamming fra tutte le parole di codice escluso $\vec{0}$, infatti:

$$\min\{d_H(c_i, c_j)\} = \min\{d_H(\vec{0}, c_i + c_j)\} = \min\{d_H(\vec{0}, c_k)\} = \min\{w(c_k)\}$$

è ancora una parola di codice *def. di peso di Hamming*

Un codice lineare a blocchi è detto **in forma sistematica** se le parole di codice sono tutte del tipo \xleftarrow{k} $\xleftarrow{m-k}$ bit di informazione bit di parità \xleftarrow{n} , ossia se:

$$I_{k \times k} \quad \text{Parità } K \times M-K \\ \vec{G} = [I_k | P] \quad \Rightarrow \quad \vec{c} = \begin{pmatrix} \vec{m} \\ \vec{K} \end{pmatrix} \vec{G} = \begin{pmatrix} \vec{m} \\ \vec{K} \end{pmatrix} \begin{bmatrix} I_k & P \end{bmatrix} = \begin{pmatrix} \vec{m} \\ \vec{m}P \end{pmatrix} = \begin{pmatrix} \vec{m} \\ \vec{m}P \end{pmatrix} = \begin{pmatrix} \vec{m} \\ \vec{g} \end{pmatrix}$$

bit di informazione ↓ ↑ parità

Possiamo allora definire la matrice di controllo di parità $I_{m-k}^T H = [P^T | I_{m-k}]$
tale che, $\forall c \in C \quad cH^T = mG H^T = \vec{0}$, infatti:

$$\begin{pmatrix} \vec{m} \\ \vec{K} \end{pmatrix} \begin{bmatrix} I_k & P \end{bmatrix} \begin{bmatrix} P \\ I_{m-k} \end{bmatrix}^T = \begin{pmatrix} \vec{m} \\ \vec{K} \end{pmatrix} \begin{bmatrix} I_k & P + P_{m-k} I_{m-k} & I_{m-k} \end{bmatrix} = \begin{pmatrix} \vec{m} \\ \vec{K} \end{pmatrix} \begin{bmatrix} P + P \\ I_{m-k} \end{bmatrix} = \begin{pmatrix} \vec{m} \\ \vec{m} \end{pmatrix} = \vec{0}$$

Capacità di rivelare errori

Sia x la parola di codice trasmessa e sia $y = x + e$ la parola di m bit ricevuta, allora ogni codice lineare a blocchi è in grado di **rivelare con certezza** fino a $d_{\min} - 1$ errori su y ($0 < w(e) < d_{\min}$), infatti:

- se $d_H(x, y) < d_{\min} \Rightarrow y$ non può essere una parola di codice, altrimenti le d_{\min} non sarebbe tale, dunque $yH^T \neq \vec{0}$ e quindi viene rivelato l'errore
- se $d_H(x, y) = d_{\min} \Rightarrow$ Detto che per definizione di d_{\min} ogni parola di codice ha almeno una distanza d_{\min} da essa, allora y potrebbe

essere una parola di codice (tra quelle distanti almeno d_{\min} da x), in tal caso $yH^T = 0$ e dunque l'errore non verrebbe rivelato

Dato $x \in C(k, m)$ allora $\exists c \in C(k, m) : d_H(x, c) = d_{\min}$

dim: esiste una parola di codice per linearità-

$$d_H(x, c) = d_H(0, x+c) = d_H(0, p) = w(p)$$

\Rightarrow posso sempre scegliere $c : p = x + c$ obbligato peso d_{\min} , al massimo piano tutti $\in 2^k c$, posso d'poi uscirne qualche $p = x + c$ con peso d_{\min} , altrimenti la d_{\min} non sarebbe tale

Capacità di correggere errori

Sia $y = x + e$ il vettore ricevuto, lo strategia di decodifica ottima è quella a massima verosimiglianza (Maximum Likelihood) e consiste nel trovare la parola di codice \hat{x} tale che, tra tutte le possibili 2^k parole, massimizza la probabilità di aver ricevuto y avendo trasmesso x : $\hat{x} = \operatorname{argmax}_{x \in C} P[y|x]$

Poiché il canale è senza memoria, gli eventi di errore sono indipendenti bit a bit, dunque $P[y|x] = \prod_{i=1}^m P[y_i|x_i]$ inoltre, poiché il canale è binario e simmetrico $P[y_i|x_i] = \begin{cases} 1-p & \text{se } y_i = x_i \\ p & \text{se } y_i \neq x_i \end{cases}$

Dato che $d_H(y, x)$ misura il numero di $i : y_i \neq x_i$, allora $m - d_H(y, x)$ misura il numero di $i : y_i = x_i$, per cui, a meno del binomiale:

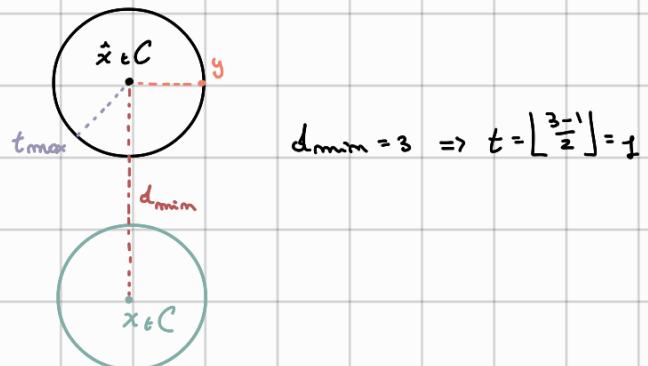
Non mi interessa
per calcolare
l'argmax

$$P[y|x] = \prod_{i=1}^m P[y_i|x_i] = p^{d_H(y, x)} (1-p)^{m-d_H(y, x)} = (1-p)^m \left(\frac{p}{1-p}\right)^{d_H(y, x)}$$

$$\Rightarrow \hat{x} = \operatorname{argmax}_{x \in C} P[y|x] = \operatorname{argmax}_{x \in C} (1-p)^m \left(\frac{p}{1-p}\right)^{d_H(y, x)} = \operatorname{argmin}_{x \in C} d_H(y, x)$$

Di conseguenza il ricevitore ML ottimo è quello che osserva al vettore in ingresso y la parola di codice x tale che $d_H(y, x)$ è minima

Il ricevitore è quindi in grado di correggere con successo tutti gli errori ϵ per cui $y = x + \epsilon$ è comunque più vicina (nel senso di Hamming) a x che a qualsiasi altro parola di codice, questi errori sono $t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor$, ossia il massimo numero per cui le sfere centrate nelle parole di codice risultano disgiunte:



$$d_{\min} = 3 \Rightarrow t = \left\lfloor \frac{3-1}{2} \right\rfloor = 1$$

Esempi di codici lineari e blocchi

Codice e ripetizione

È il codice più semplice che esista, può essere scritto in forme sistematiche come $G = \begin{bmatrix} m \\ k \end{bmatrix}$ matrici J_k], ad esempio:

$$\rightarrow \text{se } n = \frac{1}{3} \quad G = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \Rightarrow$$

$$\rightarrow \text{se } n = \frac{2}{6} \quad G = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \Rightarrow$$

\Rightarrow La domenica quest'ottica è $\frac{m}{k}$

Colici e controllo di parità

È un codice a rate fisso $R = \frac{K}{K+1}$ il quale introduce un bit di (dis)parità che vale 1 se il messaggio contiene un numero dispari di 1, 0 altrimenti.

Il vettore colonna di

$$\text{Ad esempio se } n = \frac{2}{3} \quad G = \left[\begin{array}{cc|c} 1 & 0 & 1 \\ 0 & 1 & 1 \end{array} \right]$$

\Rightarrow la somma di questi codici è sempre 2

Codice di Hammurabi

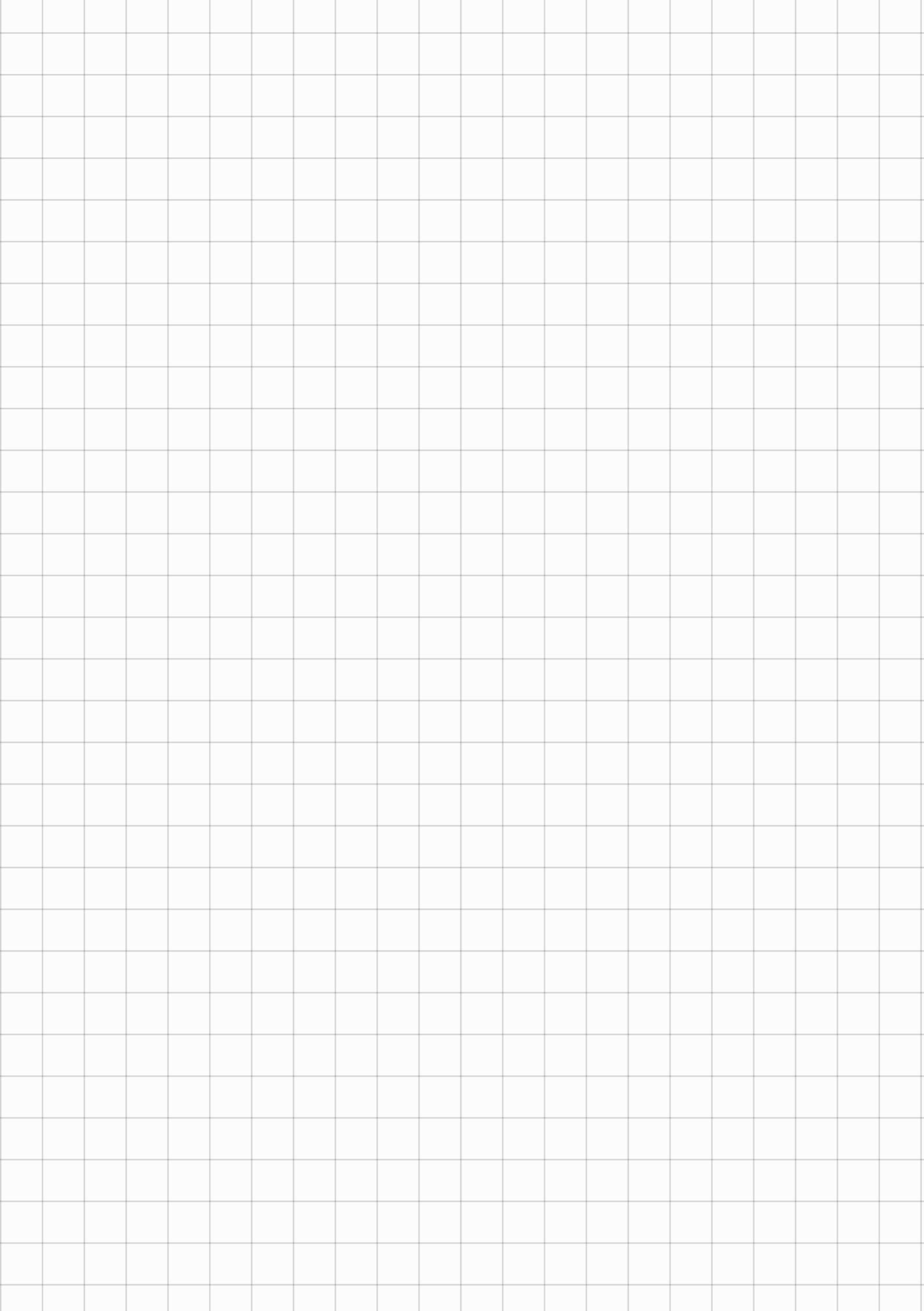
Un codice di Hamming $C_n(m)$ è un codice lineare a blocchi caratterizzato da un parametro $m \geq 2 \Rightarrow n = 2^m - 1$, $k = 2^m - m - 2$

In un codice di Hamming in forma sistematica la matrice di parità P viene costruita in modo che le colonne delle matrice controllo di parità $H = [P^T, I_{m-k}]$ corrispondono a tutti i possibili sottoinsiemi di m elementi escluso quello di tutti zeri, si può dimostrare che ogni codice di Hamming ha $d_{min} = 3$

→ se $m=2$ il codice è equivalente ad uno a ripetizione con $r = \frac{1}{3}$

→ se $m=3$ otengo $n = 2^3 - 1 = 7$, $k = 2^3 - 3 - 2 = 4$, una sua possibile coppia G, H in forma sistematica è la seguente:

$$H = \left[\begin{array}{ccc|cc} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{array} \right] \xrightarrow{\text{P}^T \quad I_{m-k}} G = \left[\begin{array}{cccc|cc} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array} \right]$$



Decodifica mediante syndrome

Un approssimazione alternativa a quella a massimo verosimiglianza su x , che produce gli stessi identici risultati, consiste nell'osservare che, dato che massimizzare $P[y|x]$ significa cercare la parola di codice x meno distante da y nel senso di Hamming, questo equivale a cercare lo x per cui, avendo ricevuto y , sono stati commessi meno errori, ossia cercare l'errore e con peso di Hamming minore tale che $y+e \in C$

$$\hookrightarrow \text{massimizzando } P[y|x] = P[x+e|x] = P[e|y+e \in C]$$

modulo 2

$$\text{allora } \hat{x} = \underset{\substack{x \in C \\ \text{ML su } x}}{\operatorname{argmax}} P[y|x] = y - \hat{e} = y + \hat{e}$$

\downarrow
MI su e

gli eventi di errore sono indipendenti bit a bit

$$\text{dove } \hat{e} = \underset{e}{\operatorname{argmax}} P[e|y+e \in C] = \underset{e \in \{y+e \in C\}}{\operatorname{argmax}} P^{\frac{w(e)}{(1-p)^{m-w(e)}}}$$

$$= \underset{e \in \{y+e \in C\}}{\operatorname{argmax}} \left(1-p\right)^{m-w(e)} \underset{\text{cost}}{\overset{w(e)}{\underset{\uparrow}{\operatorname{argmin}}}} w(e)$$

Ora da poc guardo solo quelli tali che $y+e \in C$ ma intanto devo calcularli, sono comunque 2^m operazioni

Per ora non è combiatto niente, anzi abbiamo peggiorato le cose perché siamo possuti a cercare da 2^k possibili x a 2^m possibili e ($m > k$), per poter cercare \hat{e} in modo efficiente dobbiamo prima introdurre il cosetto di **coset**

Sia $C(k,m)$ un codice lineare e sia $v \in V^m$ un vettore di m cifre binarie, si definisce coset di C individuato da v $C_v = C + v = \{x+v : x \in C\}$, ogni coset gode delle seguenti proprietà:

- Qualsiasi vettore di V^m appartiene ad un coset di $C(k,m)$
- Ciascun coset contiene 2^k elementi
- Due coset o sono coincidenti o hanno intersezione nulla
- Ci sono 2^{m-k} coset distinti

Ad esempio sia $C(2,3) = \{000, 101, 010, 111\}$, i coset di $C(2,3)$ sono:

$$C + 000 = \{000, 101, 010, 111\} = C_0$$

$$C + 001 = \{001, 100, 011, 110\} = C_1$$

$$C + 010 = \{010, 111, 000, 101\} = C_2$$

$$C + 011 = \{011, 110, 001, 100\} = C_3$$

$$C + 100 = \{100, 001, 110, 010\} = C_4$$

$$C + 101 = \{101, 000, 111, 011\} = C_5$$

$$C + 110 = \{110, 011, 100, 001\} = C_6$$

$$C + 111 = \{111, 010, 101, 000\} = C_7$$

Ritorniamo al problema iniziale:

Ho ricevuto y che so essere $x \in C$ più un qualche errore e

Non conosco né x né e e voglio stimare e per ottenere x

Dovrei prendere ogni possibile errore, vedere se sommato ad y mi dà una parola di codice, e poi, fra questi, prendere quello con peso di Hamming minore, ossia il più probabile

$$e \in C_y \Rightarrow y + e = y + c + y = c$$

Tuttavia, dato che C_y contiene per definizione tutti gli elementi che sommati ad y danno una parola di codice, allora questo contiene certamente anche il particolare e che sto cercando e dunque le sue stime diverse

semplicemente l'elemento di C_y con peso di Hamming minore
(detto coset leader) $C_y = C + y = C + x + e = C + e = C_e$ e $e \in C_y \forall e \in C_e \Rightarrow e \in C_e = C_y$

Ad esempio sia $C(2,4) = \{0000, 1011, 0101, 1110\}$ con $d_{min} = 2$ e coset

$$C_1 = C + 0000 = \{0000, 1011, 0101, 1110\}$$

$$C_2 = C + 0001 = \{0001, 1010, 0100, 1111\}$$

$$C_3 = C + 0010 = \{0010, 1001, 0111, 1100\}$$

$$C_4 = C + 1000 = \{1000, 0011, 1101, 0110\}$$

decodificare $y = [1101]$ e $y = [1111]$

$\rightarrow y = [1101] \Rightarrow C_y = C_1$ perché $y \in C_1$, allora $\hat{x} = y + \hat{e} = 1101 + 1000 = 0101$

$\rightarrow y = [1111] \Rightarrow C_y = C_2$ perché $y \in C_2$, allora $\hat{x} = y + \hat{e} = 1111 + \begin{matrix} 0001 \\ 0 \\ 0 \\ 0 \end{matrix} \rightarrow \text{Erreto non corretto!}$

Se non ce li avevo già calcolato solo C_y

Nonostante tutto questo i costi costi computazionali sono comunque troppo alti.

ML su \hat{x}

\rightarrow Se lo faccio ogni volta è $O(2^k)$ → calcolo $d_H(y, x) \forall x \in C$

\rightarrow Se lo precalcolo serve $O(2^m)$ spazio → tabella $y \rightarrow x : d_H(y, x)$ è min

ML su \hat{e}

\rightarrow Se lo faccio ogni volta è $O(2^k)$ → calcolo $(y \in \text{scelgo il coset leader})$

\rightarrow Se lo precalcolo serve $O(2^m)$ spazio → tabella $y \rightarrow \text{coset leader di } y$

Per sfruttare i coset in modo efficiente bisogna introdurre il concetto di

syndrome: si definisce syndrome di y , $s = yH^T = (x + e)H^T = xH^T + eH^T = eH^T$

$$v_1, v_2 \in C \Rightarrow v_1 + v_2 = v + c_1 + v + c_2 = c_1 + c_2 = c_3$$

Dato che $cH^T = 0 \forall c \in C$ e tutti i membri di ogni coset differiscono di una parola di codice, allora esistono 2^{m-k} syndromi associati ai 2^{m-k} coset distinti, ciascuno delle quali riconosce 2^k pattern di errore

Se precalcolassimo una tabella syndrome → coset leader allora il costo computazionale scenderebbe a:

$\rightarrow O(m \cdot (m-k))$ temporale, ossia quello di calcolare $s = yH^T$

$\rightarrow O(2^{m-k})$ spaziale, per memorizzare la tabella

Esempio

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \quad H^T = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad m=6, k=3 \Rightarrow 2^{m-k}=8 \text{ syndromi}$$

$s | \hat{e} \rightsquigarrow$ dato che $s = eH^T$ per trovare il coset leader passo andare a

001	000001
010	000010
011	010000
100	000100
101	001000
110	100000
111	010100, 100001, 001010

cercare e tale che soddisfi l'eq. sopra, partendo dagli ϵ di peso 1 ed aumentando, evitando dunque il calcolo dei coset
 → selezione la prima riga di H
 $010100, 100001, 001010 \rightarrow$ se esiste questo non riesce a correggere

$$\text{Decodifica } y = 110110 \Rightarrow s = yH^T = 011 \Rightarrow \hat{\epsilon} = 010000 \Rightarrow \hat{x} = y + \hat{\epsilon} = 100110$$

All'esame potevo calcolare solo il coset leader di questo syndrome

Per finire vediamo brevemente le probabilità di errore:

Supponendo il canale binario, simmetrico e senza memoria (le probabilità di errore sono indipendenti per ogni bit), allora la probabilità di errore sulla parola di codice decodificata è esattamente:

$$P_{w(e)} = \sum_{i=t+1}^m \binom{m}{i} p^i (1-p)^{m-i} \text{ dare } t = \left\lfloor \frac{d_{\min}-1}{2} \right\rfloor \rightsquigarrow \text{numero massimo di errori che riesce a corruggere}$$

La probabilità di errore sul bit decodificato invece non può essere calcolata esattamente perché, in presenza di un numero di errori non correggibile (dunque $> d - t$), lo decodifico stesso può introdurne ulteriori.

Dato che lo decodifico restituisce sempre una parola di codice, allora ogni volta che c'è un errore in decodifico i bit errati sono sempre almeno d_{\min} , allora possiamo stimare la probabilità di errore sul bit decodificato come:

$$P_b(e) \approx \frac{d_{\min}}{m} P_w(e) \approx \frac{d_{\min}}{m} \binom{m}{t+1} p^{t+1} (1-p)^{m-(t+1)}$$

↓
 Stima ulteriore considerando solo il caso più probabile di $P_w(e)$