

Teoria dei Codici

Davide Tonelli & Francesco Taverna

Contents

1	Introduzione	2
1.1	Ipotesi che faremo nelle nostre trattazioni	4
2	Codice a blocco	4
2.1	Codici a ripetizione	4
2.1.1	Distribuzione di Bernoulli	5
2.2	Codici a controllo di parità	7
2.3	Codice ISBN(International Standard Book Number)	8
3	Definizione di campo	9
3.1	Campo di Galois	11
3.2	Codici a blocco lineari su GF(2)	11
3.3	Proprietà G	12
3.4	Proprietà codici lineari a blocchi	12
3.5	Distanza di Hamming	13
3.6	Codici a blocco in forma sistematica	14
3.6.1	Esempio codice a ripetizione 1/3	15
3.6.2	Esempio parità 7/8	15
3.7	Codici a blocco in forma sistematica	16
3.7.1	Matrice controllo parità	17
3.7.2	Esempio codice controllo parità	17
4	Peso di Hamming e distanza minima di un codice	18
5	Rivelazione e decodifica degli errori	20
5.1	Error detection	21
5.1.1	Massima verosimiglianza	23
5.2	Error correction	25
6	Codici di Hamming	29
6.1	$C_H(2)$	30
6.2	$C_H(3)$	31
7	Decodifica codici a blocco	31
7.1	Coset	34
7.1.1	Esempio di coset	36
7.2	Esempio di decodifica con coset	37
7.3	Decodifica mediante sindrome	39
7.4	Esercizio	42
7.4.1	Svolgimento	42

7.5 Codici ciclici	43
7.5.1 Esempi	45
7.5.2 Rappresentazione algebrica	46
7.5.3 Polinomio generatore	49
7.5.4 Esempi	50
7.5.5 Polinomio generatore parte-2	51
7.5.6 Teorema fondamentale generatore di un codice ciclico	53
7.5.7 Divisione tra polinomi in GF(2)	55
7.5.8 Matrice generatrice di un codice ciclico	58
7.5.9 Controllo di parità per un codice ciclico	59
7.5.10 Esempio di calcolo di matrice generatrice e di controllo di parità	62
7.5.11 Metodo alternativo per calcolo della sindrome	63
7.5.12 Decodifica a sindrome per codici ciclici	64
7.5.13 Esempio di decodifica per codici ciclici	66
8 Codici convoluzionali	72
8.1 Introduzione	72
8.2 Generatori	75
8.2.1 Esempio di generatore	76
8.2.2 Macchina a stati	79
8.3 Diagramma a traliccio	82
8.3.1 Distanza colonna	84
8.4 Distanza libera	85
8.5 Decodifica a massima verosimiglianza	86
8.6 Algoritmo di Viterbi	89
8.6.1 Esempio di applicazione di Viterbi	92
8.6.2 Considerazioni algoritmo di Viterbi	99
9 Calcolo della probabilità di errore sulle parole di codice	101
9.1 Confronto delle prestazioni tra sistemi codificati e non	105
10 Codici Convoluzionali	109
10.1 Puncturing per codici convoluzionali	111

1 Introduzione

L'informazione può essere:

- Compressa
tolgo ridondanza
- Crittografata
- Corretta
metto ridondanza per assicurare correttezza

L'informazione po' essere compressa come:

- lossy
- lossless

Un esempio sono le compressioni dei cd. Gli mp3 sono lossy, ma l'informazione che si perde sono frequenze che nella maggior parte dei casi nemmeno sentiamo. Per gli errori abbiamo:

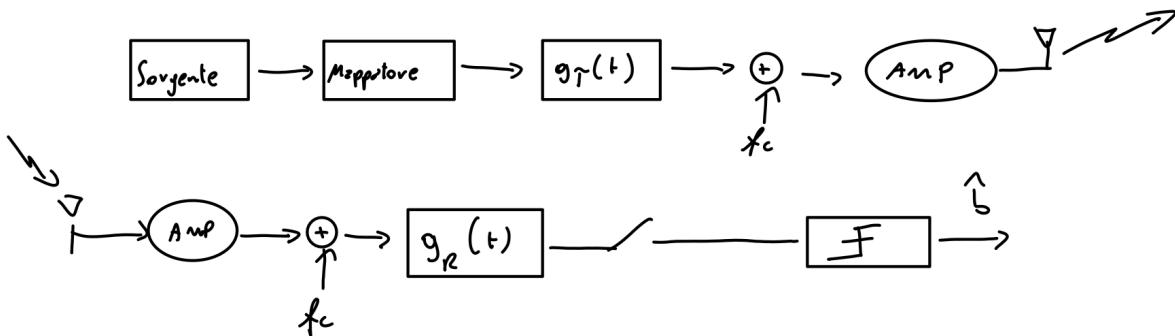
- rivelazione
Dice se c'è errore
- protezione
Dice qual'è l'errore.

Capacità di canale: La capacità di informazione che può passare al massimo in un canale in maniera affidabile.

$$C = B \log(1 + SNR)$$

- SNR = signal to noise ratio
- B = banda

Affidabile vuol dire che riuscire' sempre a trovare un codice complicato quanto ti pare, ma che garantisce la protezione perfetta dall'errore. Questo solo se non supera la capacità **C** del canale.

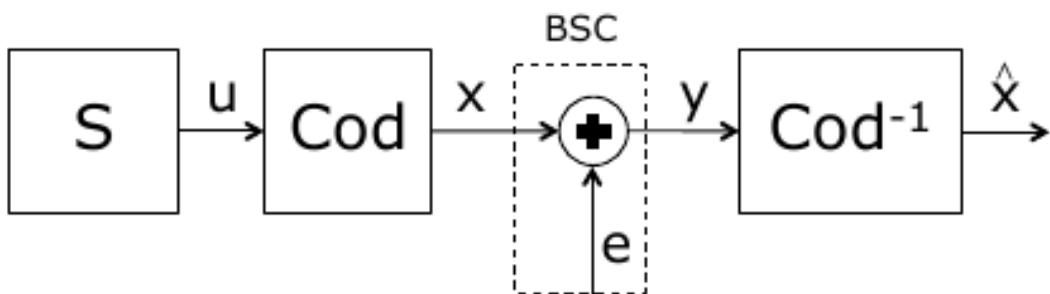


Nella teoria dei codici tutto questo si vede a un livello piu' alto. A noi interessa solo la sorgente e l'uscita del decisore.

I bit che escono dalla sorgente e in ingresso al codificatore sono identificati con u e per noi rappresentano l'**informazione**

x è il dato codificato.

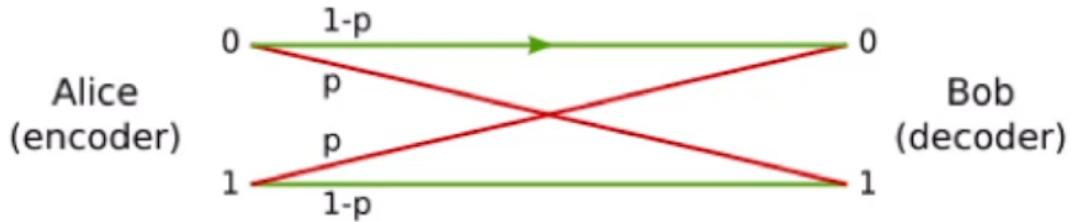
Il codificatore aggiunge informazione per togliere errori e quindi $\dim(x) > \dim(u)$. Per rappresentare gli errori nel canale introduco un vettore e che sommo a x .



Il decodificatore dovrà accorgersi degli errori e magari anche correggerli.

1.1 Ipotesi che faremo nelle nostre trattazioni

- Binary Symmetric Channel



La probabilità che un 1 diventi 0 = p . la probabilità che uno 0 diventi 1 = $1-p$. Pero' per noi $1 - p = p$ quindi $p = 0.5$.

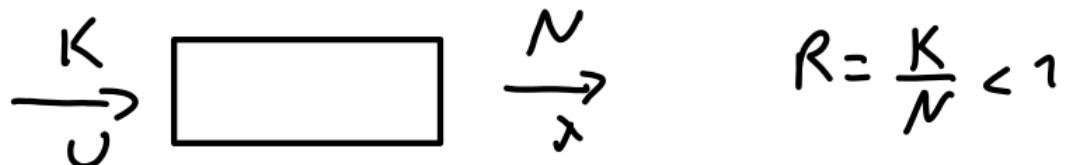
- Indipendenza statistica

$$p(a, b) = p(a) \cdot p(b)$$

Ovvero per noi vale che la probabilità di errore su un bit sia indipendente dalla probabilità di errore su un altro bit. Quindi se per esempio la probabilità di sbagliare un bit è 10^{-2} . Per noi la probabilità di sbagliarne due di fila diventa: $10^{-2} \cdot 10^{-2} = 10^{-4}$.

2 Codice a blocco

Si prende un blocco di dati e a coda di quel blocco ci si mette delle info di controllo. La prima cosa che ci interessa quando si definisce un codice è il **rate** del codice.



Siccome $N > K$ abbiamo come conseguenza che:

$$R = \frac{K}{N} < 1$$

Quando trasmetto io voglio avere il rate sempre il piu' possibile vicino a 1.

Supponiamo ora di avere 1024 parole di codice tutte differenti, ciascuna lunga 15 bit. Io so che $N = 15$. Deduco che $K = 10$ bit perché $2^{10} = 1024$. Ovvero posso decodificare su 10 bit 1024 parole diverse.

2.1 Codici a ripetizione

Possono essere usati sia per rivelazione che per protezione. Si ripete un codice $n-1$ volte. Ovvero se devo trasmettere 101 in realtà con $n = 3$ diventerà:

111000111

Per esempio con un rate di $\frac{1}{3}$ codificando un bit alla volta ogni bit viene ripetuto 3 volte. Il ricevitore il bit lo riceve per votazione. Anche detta **decisione per maggioranza**. Ovviamemente necessito che n sia dispari. Per esempio se ho $k = 1$ quindi 2¹ parole di codice con $N = 3$ ho

$$n = 3 \quad 2^k \quad \text{parole} \quad \left\{ \begin{array}{l} [000] \\ [111] \end{array} \right. \quad \sim$$

In generale siccome uso decisione a maggioranza se voglio rivelare gli errori devo avere un numero n dispari e se ho n dispari il numero massimo di errori che riesco a correggere sono $\frac{n-1}{2}$.

Se io voglio calcolare la probabilità di fare un errore su tre: Avendo come x:

- 0 0 0

Come Y con un errore posso avere:

- 1 0 0
- 0 1 0
- 0 0 1

Quindi la probabilità che riceva un errore è $(1-p)^2p$ con n(bit) = 3 e t(errori) = 1.

2.1.1 Distribuzione di Bernoulli

La formula dell'errore su un numero fisso di bit.

$$p(t, n) = \binom{n}{t} p^t (1-p)^{(n-t)} \quad (1)$$

Questa formula vuol dire che la probabilità di sbagliare t bit in una parola composta da n bit è uguale alla probabilità di sbagliare t bit + la probabilità di non sbagliare i restanti (n-t). Il coefficiente binomiale tiene conto delle permutazioni.

- p (nella formula di destra) è la probabilità di commettere un errore su un bit.

Il coefficiente binomiale è:

$$\binom{n}{t} = \frac{n!}{t!(n-t)!}$$

In generale per tenere conto di tutti i numeri di errori che possono verificarsi faccio:

$$p_{e,W} = \sum_{j=\delta+1}^N \binom{N}{j} p_{e,b}^j (1-p_{e,b})^{(N-j)} \quad (2)$$

- $\delta \rightarrow$ numero di errori correggibili
- $N \rightarrow$ numero di bit del messaggio

Nel caso col rate $\frac{1}{3}$ con 5 bit abbiamo:

$$\begin{aligned} n &= 3 \quad t = 1 \quad \binom{3}{1} = \frac{3!}{2! \cdot 1!} = 3 \\ P_r\{t=3, n=5\} &= \binom{5}{3} p^3 (1-p)^2 \\ &\binom{5}{3} = \frac{5!}{3! \cdot 2!} = 10 \end{aligned}$$

$\Pr\{\text{Codice a ripetizione con } R = \frac{1}{5} \text{ non riesce a correggere gli errori introdotti dal canale}\} =$

$$\sum_{t=3}^5 \binom{5}{t} p^t (1-p)^{5-t}$$

Questa è la forma estesa di quella formula di sopra. Questa è la probabilità complessiva che io sia riuscito a non fare errori. Quindi in un parola con 5 bit posso correggere al massimo 2 errori.

Vediamo ora con un esempio i vari casi che possono esistere in cui ho 3 errori sui 5: X:

- 0 0 0 0 0

Y:

1. 1 1 1 0 0
2. 1 1 0 1 0
3. 1 1 0 0 1
4. 1 0 1 0 1
5. 1 0 1 1 0
6. 1 0 0 1 1
7. 0 1 1 1 0
8. 0 1 1 0 1
9. 0 1 0 1 1
10. 0 0 1 1 1

Andiamo ad usare la formula 1. Ora supponiamo che $P = 10^{-2}$:

- $\Pr\{3 \text{ errori su } 5\} = 10 \cdot 10^{-2^3} (0.99)^2 = 9.8 \cdot 10^{-6}$
- $\Pr\{4 \text{ errori su } 5\} = 5 \cdot 10^{-2^4} (0.99) = 5 \cdot 10^{-8}$
- $\Pr\{5 \text{ errori su } 5\} = 1 \cdot 10^{-2^5} (0.99)^0 = 10 \cdot 10^{-10}$

Pel considerarli tutti dovremmo usare l'equazione 2, ma quello con 3 errori é decisamente piu' dominante rispetto agli altri e quindi si puo' considerare solo quello.

2.2 Codici a controllo di parità

Di solito questi codici sono a rivelazione di errore. Contano il numero di 1 presenti nella parola. Se sono pari in fondo alla parola ci metto un 1. Al ricevitore conto il numero di 1 e vedo se corrisponde. Questo rileva solo un numero dispari di errori, ma la probabilità che io faccia 1 errore é l'evento piu' probabile.

- $p(1-p)^{n-1} \rightarrow 1 \text{ errore}$
- $p^2(1-p)^{n-2} \rightarrow 2 \text{ errore}$
- $p^3(1-p)^{n-3} \rightarrow 3 \text{ errore}$

Abbiamo quindi che al messaggio m codificato come u viene affiancato un bit p di parità $\rightarrow [u,p]$. É un codice molto poco potente, pero' é anche vero che la ridondanza introdotta é molto bassa. L'esempio di codice a controllo di parità piu' semplice é quello dei codici ascii estesi. La probabilità che io faccia un errore nella trasmissione di un ascii character senza parità in un mezzo con probabilità di errore: $p_{e,b} = 10^{-8}$ é:

$$p_{e,w} = \sum_{j=1}^{11} \binom{11}{j} p_{e,b}^j (1 - p_{e,b})^{(11-j)} \approx \\ 11p_{e,b}(1 - p_{e,b})^{10} \approx \\ 11p$$

Il rate delle parole sbagliate é:

$$R_{e,w} = \frac{R_B}{11} \cdot p_{e,W} \approx \\ \left(\frac{10^7}{11} \cdot 11p \right) = 0.1 \frac{w}{s}$$

Vediamo come campia ora considerando di mettere un solo bit di parità:

$$p_{e,w} = \sum_{j=2}^{12} \binom{12}{j} p_{e,b}^j (1 - p_{e,b})^{(12-j)} \approx \\ 66p_{e,b}^2(1 - p_{e,b})^{10} \approx 66p_{e,b}^2$$

La formula la puoi approssimare prendendo solo il primo elemento. E quindi il rate

$$R_{e,w} = \frac{R_B}{12} \cdot p_{e,W} \approx \\ \left(\frac{10^7}{12} \cdot 66p_{e,b}^2 \right) = 5.5 \cdot 10^{-9} \frac{w}{s}$$

Sbaglio quindi una parola ogni $T_{e,w} = \frac{1}{R_{e,w}} = 1.82 \cdot 10^8 s$ ovvero una parola ogni circa 6 anni. Introducendo il bit di parità ora sbaglio quando faccio 2 errori. I pari piu' grandi li posso ignorare sempre perche' trascurabili.

Notiamo inoltre che é piu' probabile che occorra un numero di errori dispari rispetto a un numero pari. Tenendo inoltre conto come dimostrato qui sopra che é piu' probabile che ci sia un numero minore di errori otteniamo che seppur semplice questa codifica é molto efficace.

2.3 Codice ISBN(International Standard Book Number)

Usa alfabeto decimale e non binario. Siccome codifica su 11 cifre abbiamo che usa come caratteri: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, x]. Si calcola la grandezza z che è propedeutica per il controllo come:

$$z = \text{mod}(S, 11) \quad (3)$$

$$S = \sum_{j=1}^9 (11 - j)x(j) \quad (4)$$

- $x(j)$ è la cifra j-esima

La cifra di controllo di parità è il complemento a 11 di z

$$x(10) = \text{mod}(11 - z, 11) \quad (5)$$

Quindi con $x(10)$ si indica la decima cifra del codice ISBN.

- $x = [u, x(10)]$
- Il pattern di errore che rileva è o un numero sbagliato o due numeri invertiti.

21/03

► Quando un dispositivo legge il codice ISBN, acquisisce la parola $y = [y(1), y(2), \dots, y(10)]$.

1. Per verificare che il codice sia corretto, il dispositivo calcola

$$\text{mod}(S', 11) \text{ con } S' = \sum_{j=1}^{10} (11 - j)y(j),$$

2. Assumendo che non ci siano errori su $x(10)$, si ha

$$\begin{aligned} \text{mod}(S', 11) &= \text{mod} \left(\sum_{j=1}^9 (11 - j)y(j) + \text{mod}(11 - z, 11), 11 \right) \\ &= \text{mod} \left(\sum_{j=1}^9 (11 - j)y(j) + \left(11 - \sum_{j=1}^9 (11 - j)x(j) \right), 11 \right) \\ &= \text{mod} \left(\sum_{j=1}^9 (11 - j)(y(j) - x(j)), 11 \right) \end{aligned}$$

Un esempio di codici a blocco: codice ISBN

- ▶ Se non ci sono errori si ha $y = x$ e quindi $\text{mod}(S', 11) = 0$.
- ▶ Il codice è in grado di rivelare tutti gli errori singoli
Sia $e(k)$ l'errore in posizione k , $y(k) = x(k) + e(k)$

$$\begin{aligned}\text{mod } (S', 11) &= \text{mod}((y(k) - x(k))(11 - k), 11) \\ &\quad + \text{mod}(e(k)(11 - k), 11) \neq 0\end{aligned}$$

- ▶ Il codice è in grado di rivelare tutti i casi in cui ci sia uno scambio di due cifre del codice. Siano k_1 e k_2 le 2 posizioni scambiate

$$\begin{aligned}\text{mod } (S', 11) &= \text{mod}((x(k_2) - x(k_1))(11 - k_1) + (x(k_1) - x(k_2))(11 - k_2), 11) \\ &= \text{mod}((x(k_2) - x(k_1))(k_2 - k_1), 11) \neq 0\end{aligned}$$

Può però accadere che in presenza di più di un errore sulle cifre questo non venga rilevato poiché abbiamo una somatoria all'interno del modulo.

3 Definizione di campo

F è un insieme di numeri

- Un campo è una struttura composta da un insieme non vuoto F e da due operazioni binarie *interne*: *somma* e *prodotto*. Per ogni $\alpha, \beta, \gamma \in F$ vale

Somma

$$\alpha + \beta \in F$$

$$(\alpha + \beta) + \gamma = \alpha + (\beta + \gamma)$$

$$\alpha + \beta = \beta + \alpha$$

$$0 \in F, \alpha + 0 = \alpha, \alpha - \alpha = 0$$

Prodotto

(2)

$$\alpha * \beta \in F$$

$$(\alpha * \beta) * \gamma = \alpha * (\beta * \gamma)$$

$$\alpha * \beta = \beta * \alpha$$

$$1 \in F, \alpha * 1 = \alpha, \forall \alpha \neq 0 \alpha * \alpha^{-1} = 1$$

$$\alpha * (\beta + \gamma) = \alpha * \beta + \alpha * \gamma$$

Le proprietà della somma sono in ordine nella foto:

- chiusa
- associativa
- commutativa
- elemento neutro
- elemento opposto

Le proprietà del prodotto sono in ordine nella foto:

- interno o chiuso
- associativo
- commutativo
- elemento neutro
- elemento inverso
- distributiva

3.1 Campo di Galois

- ▶ Un campo di Galois $GF(q)$ è un campo con un *numero finito* q di elementi.
- ▶ $GF(2)$ è il campo definito su $\{0, 1\}$ con somma modulo 2 (“XOR”) e prodotto modulo 2 (“AND”).
- ▶ Una volta definito $GF(2)$, si può costruire lo spazio vettoriale $\mathcal{V}_n = GF(2)^n$, lo spazio di tutte i possibili 2^n vettori di n cifre binarie su cui valgono le operazioni definite per $GF(2)$.

$GF(2)$ è il campo di dimensione minima per trovare numeri che possano rispettare le proprietà elencate. Si puo' verificare che tutte le proprietà precedentemente citate sono rispettate dalle due operazioni di somma e prodotto indicate sopra.

3.2 Codici a blocco lineari su $GF(2)$

Si definisce lineare:

$$f(\alpha x_1 + \beta x_2) = \alpha f(x_1) + \beta f(x_2) \quad (6)$$

- ▶ Sia $\mathbf{u} = [u_1, u_2, \dots, u_k]$ una generica parola di k cifre binarie. Il codice a blocco lineare $\mathcal{C}(k, n) \subset \mathcal{V}_n$ è l'insieme delle 2^k parole $\mathbf{x} = [x_1, x_2, \dots, x_n]$ di n cifre binarie ottenute con la trasformazione lineare

$$\mathbf{x} = \mathbf{u}\mathbf{G} \quad (3)$$

dove \mathbf{G} è una matrice $k \times n$ di cifre binarie.

- ▶ \mathbf{G} è la *matrice generatrice* del codice.

Ricorda che $n > k$ perché la codifica introduce ridondanza. Nota inoltre che siccome il sistema è specchiato rispetto a quello classico x e u sono vettori riga e non colonna.

3.3 Proprietà G

- ▶ Siano \mathbf{g}_i ($i = 1, 2, \dots, k$) le righe di \mathbf{G} , \mathbf{x} è la combinazione lineare delle righe \mathbf{g}_i .

$$\mathbf{x} = \sum_{i=1}^k u_i \mathbf{g}_i \quad (4)$$

- ▶ Perché ci siano 2^k parole di codice distinte è necessario che \mathbf{G} abbia rango $k \implies$ le righe di \mathbf{G} sono linearmente indipendenti e costituiscono una *base per il sottospazio vettoriale* $\mathcal{C} \subset \mathcal{V}_n$.

La matrice ha rango K che é il massimo possibile ovvero essendo non quadrata il minimo tra k e n essendo \mathbf{G} $k \times n$.

- $\mathbf{G} = k \times n$
- $\mathbf{u} = 1 \times k$
- $\mathbf{x} = 1 \times n$

3.4 Proprietà codici lineari a blocchi

- ▶ Alcune semplici proprietà derivano direttamente dalla linearità dei codici:
 1. Ogni parola di codice è una combinazione lineare di righe della matrice generatrice.
 2. Il codice a blocchi è costituito da tutte le possibili combinazioni delle righe della matrice generatrice.
 3. La somma di due parole di codice è ancora una parola di codice.
 4. La n -pla di tutti zeri è sempre una parola di codice.
 5. Se \mathbf{x} è una parola di codice, anche $-\mathbf{x}$ è una parola di codice.

Noi abbiamo 2^k valori per parole, esse vengono mappate in parole codificate su n bit quindi 2^n possibilità, pero' le parole di partenza possono identificare solo 2^k valori che é un sottoinsieme proprio di 2^n . Dunque il ricevitore é in grado di verificare se la parola ricevuta sia corretta o no. Per esempio con un codice a ridondanza abbiamo:

- $k=7 \rightarrow 128$

- $n=8 \rightarrow 256$

Quindi metà sono non valide. Quella metà è quindi scartata perché quelle con numero di 1 dispari sono non valide.

3.5 Distanza di Hamming

Se trasmetti una parola di n cifre e il canale introduce un po' di errori al ricevente. Mi piacerebbe associare la parola di codice ricevuta a quella valida più vicina. Bisogna quindi introdurre il concetto di distanza. La distanza tra due parole di codice è data dal numero di posizioni diverse delle due parole e questa si definisce come distanza di Hamming. Esempio:

$$\left. \begin{array}{l} x_1 = 010 \\ x_2 = 001 \end{array} \right\} \text{distanza di Hamming} = 2$$

- ▶ La distanza di Hamming $d(\mathbf{x}_1, \mathbf{x}_2)$ tra due vettori di n elementi \mathbf{x}_1 e \mathbf{x}_2 è il numero di posizioni in cui le due parole sono diverse tra loro.
- ▶ La distanza di Hamming è una metrica.
 1. $d(\mathbf{x}_1, \mathbf{x}_2) \geq 0$
 2. $d(\mathbf{x}_1, \mathbf{x}_2) = 0 \Leftrightarrow \mathbf{x}_1 = \mathbf{x}_2$
 3. $d(\mathbf{x}_1, \mathbf{x}_2) = d(\mathbf{x}_2, \mathbf{x}_1)$
 4. $d(\mathbf{x}_1, \mathbf{x}_3) \leq d(\mathbf{x}_1, \mathbf{x}_2) + d(\mathbf{x}_2, \mathbf{x}_3)$
- ▶ Il peso di Hamming di un vettore $\mathbf{x}_0 \in \mathcal{V}_n$ è $w(\mathbf{x}_0) = d(\mathbf{x}_0, \mathbf{0}_n)$
- ▶ La *distanza minima* di un codice \mathcal{C} è la minima distanza di Hamming calcolata fra tutte le possibili parole che appartengono a \mathcal{C} .

- La 4 è la diseguaglianza triangolare
- Il peso di Hamming è quindi il numero di 1 presenti all'interno del codice.

Se vogliamo inviare un codice noi vogliamo che la distanza minima della codifica sia grande perché al ricevitore risulta più facile associare la parola corretta a quella ricevuta in caso di errori.

Esempio

- ($d_{\min} = 5$)
- Insieme di codici: 0 5 10 20 30
- Invio 5 e ricevo 4

Comunque riesco a capire che ho ricevuto 5

- ($d_{\min} = 1$)

- Insieme di codici: 1 2 3 4 5 600 700 800
- Invio 5 e ricevo 4

Non riesco a capire che ho inviato 5 perché va più vicino e in questo caso si sovrappone a un altro.
Per questo è importante che la d_{\min} sia grande.

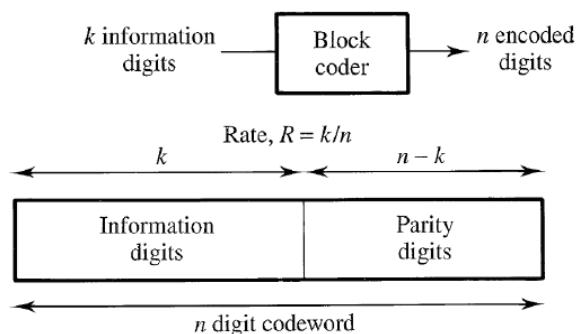
3.6 Codici a blocco in forma sistematica

Codici a blocco in forma sistematica

- Quando il codice è in *forma sistematica* la matrice generatrice del codice ha la seguente forma

$$\mathbf{G} = [\mathbf{I}_k, \mathbf{P}] \quad (5)$$

- La matrice \mathbf{P} , di dimensioni $k \times (n - k)$ è la *matrice di parità*.



- La matrice identica $k \times k$
- In \mathbf{P} le prime k cifre sono esattamente \mathbf{u} (il messaggio inviato), le successive $n - k$ sono una combinazione delle prime.

3.6.1 Esempio codice a ripetizione 1/3

- ▶ Codice a ripetizione $R = 1/3$

Bit in ingresso	Parola codificata
1	[111]
0	[000]

- ▶ La matrice generatrice del codice è

$$\mathbf{G} = [111] \quad (6)$$

- ▶ La distanza minima del codice è $d_{min} = 3$.

Per vedere che sia vero che $\mathbf{G} = [1 1 1]$ vediamo che:

- $u: 0 \rightarrow 0 \cdot [1 1 1] = [0 0 0]$
- $u: 1 \rightarrow 1 \cdot [1 1 1] = [1 1 1]$

3.6.2 Esempio parità 7/8

- ▶ Codice a controllo di parità $R = 7/8$. Ogni 7 bit ne aggiunge uno di controllo di parità: 1 se il numero di '1' è dispari, 0 se il numero di '1' è pari.
- ▶ La matrice generatrice del codice è

$$\mathbf{G} = [\mathbf{I}_7, \mathbf{1}_7] \quad (7)$$

- ▶ il prodotto $\mathbf{u}\mathbf{1}_7 = \sum_{i=1}^7 u_i$ può essere scritto come una somma modulo 2 e quindi vale 0 se il numero di '1' è pari e 1 altrimenti.
- ▶ La distanza minima del codice è $d_{min} = 2$. Dimostrazione.

$d_{min} = 2$ perché se cambio un numero deve cambiare anche quello di parità. Un altro modo di vederla è che non esistono parole con numero dispari di 1, perché se ho parole con numero dispari di 1 il bit di parità diventa 1 e quindi la somma torna pari.

3.7 Codici a blocco in forma sistematica

Vediamo ora come decodificare i codici.

Definizione: Due codici lineari $\mathcal{C}_1(k, n)$ e $\mathcal{C}_2(k, n)$ in $GF(2)$ sono *equivalenti* se uno è ottenuto dall'altro attraverso una permutazione delle posizioni del codice;

Teorema 1: Due matrici generatici \mathbf{G}_1 and \mathbf{G}_2 in $GF(2)$ generano due codici equivalenti se una può essere ottenuta dall'altra da una sequenza di operazioni di questo tipo:

1. Permutazione delle righe;
2. Combinazione lineare di righe;
3. Permutazione delle colonne.

Teorema 2: Qualsiasi codice lineare a blocchi è equivalente ad un codice in forma sistematica.

- ▶ Dato il sottospazio $\mathcal{C} \subset \mathcal{V}_n$ di dimensione k esiste un sottospazio ortogonale (null space) $\mathcal{C}^\perp \subset \mathcal{V}_n$ di dimensione $n - k$, definito dalla matrice \mathbf{H} di dimensioni $n - k \times n$ tale che

$$\mathbf{GH}^T = \mathbf{0}_{k, n-k} \quad (8)$$

- ▶ La base di \mathcal{C}^\perp è costituita dalle $n - k$ righe della matrice \mathbf{H} , per cui ogni elemento $\mathbf{t} \in \mathcal{C}^\perp$ può essere rappresentato

$$\mathbf{t} = \mathbf{v}\mathbf{H} = \sum_{i=1}^{n-k} v_i \mathbf{h}_i \quad (9)$$

- ▶ Per ogni $\mathbf{x} \in \mathcal{C}$ e per ogni $\mathbf{t} \in \mathcal{C}^\perp$ si ha

$$\mathbf{x}\mathbf{t}^T = \mathbf{u}\mathbf{G}\mathbf{H}^T\mathbf{v}^T = 0 \quad (10)$$

3.7.1 Matrice controllo parità

- ▶ La matrice \mathbf{H} è la *matrice di controllo di parità* del codice.
- ▶ Per costruzione, per ciascun $\mathbf{x} \in \mathcal{C}$ vale

$$\mathbf{x}\mathbf{H}^T = \mathbf{u}\mathbf{G}\mathbf{H}^T = \mathbf{0}. \quad (11)$$

- ▶ La matrice la matrice di controllo di parità non è unica. Se \mathbf{G} è sistematica si può utilizzare la relazione

$$[\mathbf{A}, \mathbf{B}] \begin{bmatrix} \mathbf{C} \\ \mathbf{D} \end{bmatrix} = \mathbf{AC} + \mathbf{BD} \quad (12)$$

per trovare

$$\mathbf{H} = [\mathbf{P}^T, \mathbf{I}_{n-k}]. \quad (13)$$

- $\mathbf{A} = k \times k$
- $\mathbf{B} = k \times (n-k)$
- $\mathbf{C} = k \times (n-k)$
- $\mathbf{D} = (n-k) \times (n-k)$
- $[\mathbf{A}, \mathbf{B}] = k \times n$

3.7.2 Esempio codice controllo parità

- ▶ Per il codice a ripetizione $R = 1/3$ si ha $k = 1, n = 3$ e $n - k = 2$, per cui la matrice la matrice di controllo di parità è

$$\mathbf{H} = [\mathbf{P}^T, \mathbf{I}_2] = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}. \quad (14)$$

- ▶ Per il codice a controllo di parità $R = 7/8$ si ha $k = 7, n = 8$ e $n - k = 1$, per cui la matrice la matrice di controllo di parità è

$$\mathbf{H} = [\mathbf{P}^T, \mathbf{I}_1] = \mathbf{1}_8^T. \quad (15)$$

4 Peso di Hamming e distanza minima di un codice

La d_{\min} non dipende dalla parola, ma dal codice. Ovvero una parola ha lo stesso insieme delle distanze rispetto a tutte le altre parole. Se vale cio' tanto vale calcolare la d_{\min} dalla parola con tutti 0 che é il peso di Hamming.

Peso di Hamming e distanza minima di un codice

- ▶ Il peso di Hamming $w(\mathbf{x})$ di una parola di codice \mathbf{x} è costituito dal numero di '1' presenti in \mathbf{x} .
- ▶ Il peso di una parola di codice $\mathbf{x} \in \mathcal{C}(k, n)$ è la sua distanza dalla n -upla di tutti '0'

$$w(\mathbf{x}) = d_H(\mathbf{x}, \mathbf{0}_{1,n}) \quad (2)$$

- ▶ Per i codici lineari vale la proprietà che ciascuna parola di codice ha lo stesso insieme di distanze dalle altre parole di codice.
- ▶ La distanza minima di un codice si può calcolare a partire da una qualsiasi parola di codice.

Per un qualsiasi codice a blocco lineare posso calcolare la d_{\min} con minimia distanza di Hamming.

Peso di Hamming e distanza minima di un codice

Teorema 3: La distanza minima del codice a blocco $\mathcal{C}(k, n)$ si può calcolare come il peso di Hamming minimo tra tutte le parole di codice

$$\begin{aligned} d_{min}(\mathcal{C}) &= \min_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{C}} d_H(\mathbf{x}_i, \mathbf{x}_j) \\ &= \min_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{C}} d_H(\mathbf{x}_i + \mathbf{x}_j, \mathbf{x}_j + \mathbf{x}_j) = \min_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{C}} d_H(\mathbf{x}_i + \mathbf{x}_j, \mathbf{0}_{1,n}) \\ &\quad [\mathbf{x}_i + \mathbf{x}_j \in \mathcal{C}(k, n)] \\ &= \min_{\mathbf{x}_i \in \mathcal{C}} w(\mathbf{x}_i) \end{aligned} \tag{3}$$



- Se sommo due parole uguali ottengo \emptyset . Ricordiamo che la somma è uno XOR.
- Se sommo due parole di codice ottengo un'altra parola di codice, ovvero:

$$\begin{aligned} x_i &= u_i G \\ x_j &= u_j G \\ x_i + x_j &= (u_i + u_j)G \end{aligned}$$

$u_i + u_j$ si puo' scrivere come u_k .

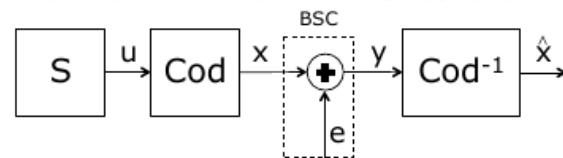
5 Rivelazione e decodifica degli errori

Rivelazione e decodifica degli errori

- ▶ Su un canale BSC senza memoria, la n -upla \mathbf{y} a valle del decisore può essere rappresentata

$$\mathbf{y} = \mathbf{x} + \mathbf{e} \quad (4)$$

dove \mathbf{e} è il vettore di errori introdotto dal canale.



- ▶ Se il canale non introduce errori $\Rightarrow \mathbf{e} = \mathbf{0}_{1,n}$.

5.1 Error detection

Capacità di rivelare errori (*error detection*)

Sia \mathbf{x} la parola di codice trasmessa e $\mathbf{y} = \mathbf{x} + \mathbf{e}$ la corrispondente sequenza di n bit ricevuta. Supponiamo che il canale introduca un certo numero di errori, i.e. $w(\mathbf{e}) > 0$.

- ▶ Se \mathbf{y} non è una parola di codice, si è verificato un errore *rivelabile*;
- ▶ Se \mathbf{y} è una parola di codice ma non quella trasmessa, si è verificato un errore *non rivelabile* ($w(\mathbf{e}) \geq d_{min}$).

- $w \rightarrow$ peso di Hamming
- i.e. \rightarrow id est (questo é)[this is]{questos estes}

Capacità di rivelare errori (*error detection*)

Il codice $\mathcal{C}(k, n)$ rivela un errore quando la parola ricevuta $\mathbf{y} \notin \mathcal{C}(k, n)$.

Teorema 4: Il codice $\mathcal{C}(k, n)$ è in grado di rivelare con certezza fino a $d_{min} - 1$ errori.

- ▶ Se $d_H(\mathbf{x}, \mathbf{y}) < d_{min} \implies \mathbf{y}$ non può essere una parola di codice, perché altrimenti vorrebbe dire che esistono due parole di codice la cui distanza è minore di d_{min} .
- ▶ Viceversa, se $d_H(\mathbf{x}, \mathbf{y}) = d_{min} \implies$ esiste almeno una parola di codice $\mathbf{c} \in \mathcal{C}(k, n), \mathbf{c} \neq \mathbf{x}$ tale che $d_H(\mathbf{x}, \mathbf{c}) = d_{min}$ e se $\mathbf{y} = \mathbf{c}$, l'errore non può essere rivelato.

5.1.1 Massima verosimiglianza

Strategia di decodifica a massima verosimiglianza

- ▶ Sia \mathbf{y} il vettore ricevuto a seguito della trasmissione su BSC, la strategia di decodifica a massima verosimiglianza (*ML, maximum likelihood*) consiste nel trovare il vettore $\hat{\mathbf{x}}$ che, fra tutte le 2^k possibili parole di codice \mathbf{x} , massimizza la probabilità condizionata $P(\mathbf{y}|\mathbf{x})$, ie

$$\hat{\mathbf{x}} = \arg \max_{\mathbf{x} \in \mathcal{C}(k,n)} P(\mathbf{y}|\mathbf{x}) \quad (5)$$

- ▶ Poichè gli eventi di errore sono indipendenti da bit a bit, si può scrivere la probabilità condizionata come il prodotto delle probabilità condizionate ottenute per ciascun bit trasmesso

$$P(\mathbf{y}|\mathbf{x}) = \prod_{\ell=1}^n P(y_\ell|x_\ell) \quad (6)$$



- $P(|)$ → probabilità condizionata = probabilità che avvenga y condizionatamente a (noto se è avvenuto) x .
- P → probabilità che accada un evento

Quello che io voglio fare è condizionare di ricavare x ricevuto y .

A me non interessa il valore specifico della parola condizionata, a me interessa il vettore x che massimizza la probabilità.

Strategia di decodifica a massima verosimiglianza

- ▶ Poiché siamo in GF(2), la probabilità $P(y_\ell|x_\ell)$ può assumere due soli valori

$$P(y_\ell|x_\ell) = \begin{cases} 1-p & \text{if } P(y_\ell = x_\ell|x_\ell) \\ p & \text{if } P(y_\ell \neq x_\ell|x_\ell) \end{cases}. \quad (7)$$

- ▶ La distanza di Hamming $d_H(\mathbf{x}, \mathbf{y})$ misura il numero di posizioni diverse tra \mathbf{x} e \mathbf{y} e quindi $n - d_H(\mathbf{x}, \mathbf{y})$ misura il numero posizioni uguali tra \mathbf{x} e \mathbf{y} .
- ▶ La probabilità $P(\mathbf{y}|\mathbf{x})$ si calcola

$$P(\mathbf{y}|\mathbf{x}) = p^{d_H(\mathbf{x}, \mathbf{y})}(1-p)^{n-d_H(\mathbf{x}, \mathbf{y})} = (1-p)^n \left(\frac{p}{1-p}\right)^{d_H(\mathbf{x}, \mathbf{y})} \quad (8)$$

◀ □ ▶ ⏪ ⏩ ⏴ ⏵ ⏹ ⏸ ⏹ ⏺ ⏻ ⏼ ⏽ ⏾ ⏿

1. Esempio

$$\begin{aligned} C &= \{0\ 0\ 0, 1\ 0\ 1, 0\ 1\ 0, 1\ 1\ 1\} \\ R &= \frac{k}{n} = \frac{2}{3} \quad 2^k = 4 \rightarrow k = 2 \\ y &= (0, 0, 0) \quad P(y|k) \end{aligned}$$

Introducendo ora dei numeri:

- $P = 10^{-1}$
- $1-P = 0.9$
- $y = [0\ 0\ 0]$

$$\begin{aligned} P(y|x=[0\ 0\ 0]) &= (1-p)(1-p)(1-p) = (1-p)^3 = 0.729 \\ P(y|x=[0\ 1\ 0]) &= (1-p)p(1-p) = p(1-p)^2 = 0.081 \\ P(y|x=[1\ 1\ 1]) &= ppp = p^3 = 0.001 \end{aligned}$$

5.2 Error correction

Decisione a massima verosimiglianza

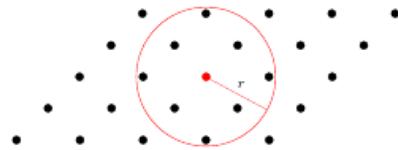
La parola di codice decisa $\hat{\mathbf{x}}$ è quella che minimizza la distanza dalla parola \mathbf{y} ricevuta

$$\hat{\mathbf{x}} = \arg \max_{\mathbf{x} \in \mathcal{C}(k,n)} P(\mathbf{y}|\mathbf{x}) = \arg \min_{\mathbf{x} \in \mathcal{C}(k,n)} d_H(\mathbf{y}, \mathbf{x}) \quad (9)$$

- ▶ Il ricevitore ML ottimo è il ricevitore a *distanza minima*: il ricevitore che associa alla sequenza di n bit ricevuta \mathbf{y} , la parola di codice \mathbf{x} che minimizza la $d_H(\mathbf{x}, \mathbf{y})$.
- ▶ Il ricevitore ML è in grado di correggere *con successo* tutti quegli errori \mathbf{e} per cui la parola ricevuta $\mathbf{y} = \mathbf{x} + \mathbf{e}$ è comunque più vicina alla parola trasmessa \mathbf{x} che a qualsiasi altra parola del codice.

Decisione a massima verosimiglianza

- ▶ Per ogni vettore $\mathbf{v} \in \mathcal{V}_n$ e un raggio r esiste una ‘sfera’ di raggio r i cui elementi sono tutti quei vettori in \mathcal{V}_n che hanno distanza di Hamming da \mathbf{v} minore o uguale a r .



- ▶ Assumendo di adottare un ricevitore ML, il numero massimo t di errori che il codice $\mathcal{C}(k, n)$ è in grado di correggere è il massimo raggio t per cui le sfere centrate nelle parole di codice di $\mathcal{C}(k, n)$ sono tutte tra loro disgiunte.

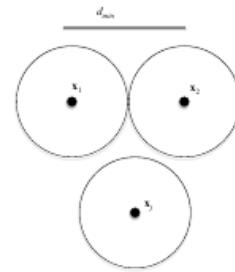
Capacità di correggere errori (*error correction*)

Teorema 5: Un codice lineare a blocco può correggere fino a $t_{max} = \lfloor \frac{d_{min}-1}{2} \rfloor$ errori, i.e. $2t_{max} + 1 \leq d_{min} \leq 2t_{max} + 2$.

La condizione per cui le sfere di raggio t che circondano le parole di codice siano disgiunte è che

$$2t_{max} < d_{min} \implies t_{max} < d_{min}/2.$$

Altrimenti, se fosse $2t_{max} \geq d_{min}$ ci sarebbero almeno due parole \mathbf{x}_1 e \mathbf{x}_2 la cui distanza $d(\mathbf{x}_1, \mathbf{x}_2) = d_{min} \leq 2t_{max}$ e le due sfere di raggio t avrebbero almeno un punto in comune.



Capacità di correggere errori (*error correction*)

Consideriamo il codice $\mathcal{C}(k, n)$ che ha una certa d_{min} e t_{max} tale che $2t_{max} + 1 \leq d_{min}$. Sia $\mathbf{x} \in \mathcal{C}(k, n)$ la parola trasmessa, $\mathbf{y} = \mathbf{x} + \mathbf{e}$ la corrispondente sequenza di n bit ricevuta e $\mathbf{c} \in \mathcal{C}(k, n)$ un'altra generica parola di codice.

Grazie alla diseguaglianza triangolare si ha

$$d_H(\mathbf{x}, \mathbf{y}) + d_H(\mathbf{c}, \mathbf{y}) \geq d_H(\mathbf{x}, \mathbf{c}) \implies d_H(\mathbf{c}, \mathbf{y}) \geq d_H(\mathbf{x}, \mathbf{c}) - d_H(\mathbf{x}, \mathbf{y}) \quad (10)$$

Per ipotesi si ha anche

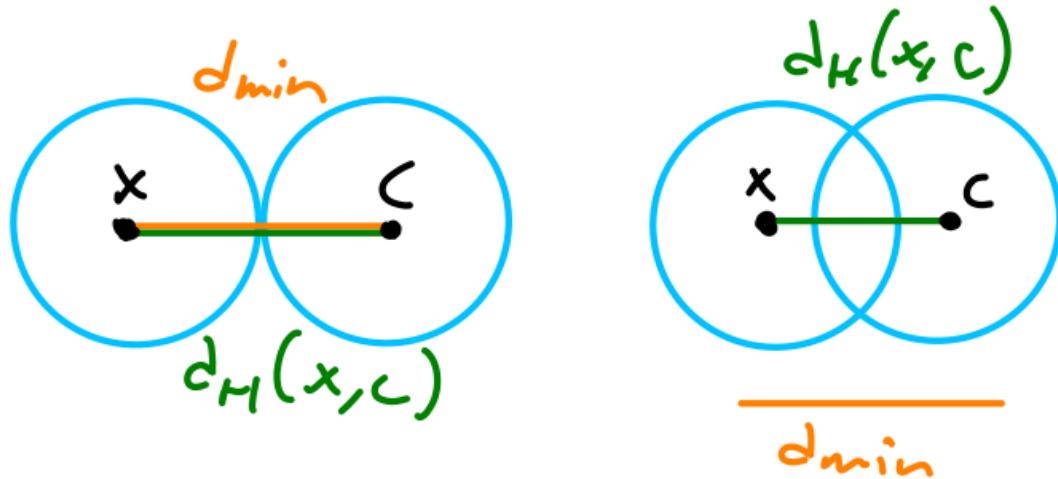
$$d_H(\mathbf{x}, \mathbf{c}) \geq d_{min} \geq 2t_{max} + 1 \quad (11)$$

Supponiamo che il canale introduca un certo numero di errori $t \leq t_{max}$, così da avere $d_H(\mathbf{x}, \mathbf{y}) = t$.

$$d_H(\mathbf{c}, \mathbf{y}) \geq 2t_{max} + 1 - t > t_{max} \geq t = d_H(\mathbf{x}, \mathbf{y}) \quad (12)$$



Per meglio spiegare d_{min} e $d_H(\mathbf{x}, \mathbf{c})$ si riporta questo disegno dove vediamo a sinistra che $d_{min} = d_H(\mathbf{x}, \mathbf{c})$, mentre a destra $d_{min} > d_H(\mathbf{x}, \mathbf{c})$. Quindi a destra non si riesce più a distinguere se un codice ricevuto sia \mathbf{x} o \mathbf{c} nel caso in cui vada a cadere nell'intersezione, mentre a sinistra in qualsiasi punto cada si riesce a distinguere.



6 Codici di Hamming

I codici di Hamming: definizione e proprietà

- ▶ I codici di Hamming sono definiti a partire da un parametro $m \geq 2$

$$n = 2^m - 1, k = 2^m - m - 1 \quad (13)$$

- ▶ Poiché la matrice di controllo di parità \mathbf{H} ha dimensione $(n - k) \times n$, per i codici di Hamming la matrice \mathbf{H} ha dimensione $m \times (2^m - 1)$.
- ▶ Per un codice di Hamming sistematico, la matrice di parità \mathbf{P} viene costruita così che le colonne di $\mathbf{H} = [\mathbf{P}^T, \mathbf{I}_{n-k}]$ siano tutte le possibili $2^m - 1$ combinazioni (escluso l' n -upla di tutti 0) di m bit.
- ▶ La distanza minima di un qualsiasi codice di Hamming $\mathcal{C}_H(m)$ è $d_{min}(\mathcal{C}_H(m)) = 3$. Dimostrazione.

- Per un esempio esplicativo del terzo punto vedi 6.2

Ricordiamo che le colonne sono tutte le possibili parole di codice valide e quindi la dimostrazione che 3 e' il numero minimo di colonne che combinate tra loro da 0:

- 1 no perche' non c'e' il vettore con tutti 0
- 2 no perche' sono tutte indipendenti
- 3 e' il numero con cui si riesce a farle combinare in vettore di tutti 0.

6.1 $C_H(2)$

I codici di Hamming: il codice $\mathcal{C}_H(2)$

- ▶ il codice a ripetizione $R = 1/3$ è il codice di Hamming con $m = 2$ con $n = 2^2 - 1 = 3$, e $k = 2^2 - 2 - 1 = 1$.
- ▶ La matrice di controllo di parità per il codice a ripetizione con $R = 1/3$ è

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad (14)$$

- ▶ Poiché la matrice \mathbf{H} ha 2 righe e 3 colonne e rappresenta tutte le possibili combinazioni di 2 bit (esclusa la n -upla di tutti 0) $\implies \mathbf{H}$ è la matrice di controllo di parità per il codice di Hamming $\mathcal{C}_H(2)$.

6.2 $C_H(3)$

I codici di Hamming: il codice $C_H(3)$

- Se $m = 3 \implies n = 7, k = 4$. La matrice di controllo di parità \mathbf{H} per $C_H(3)$ ha 3 righe e 7 colonne. In forma sistematica una possibile coppia \mathbf{H} e \mathbf{G} è

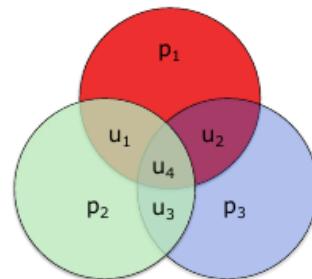
$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \implies \mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

- Si fanno controlli di parità su combinazioni diverse di bit di ingresso.

$$p_1 = u_1 + u_2 + u_4$$

$$p_2 = u_1 + u_3 + u_4$$

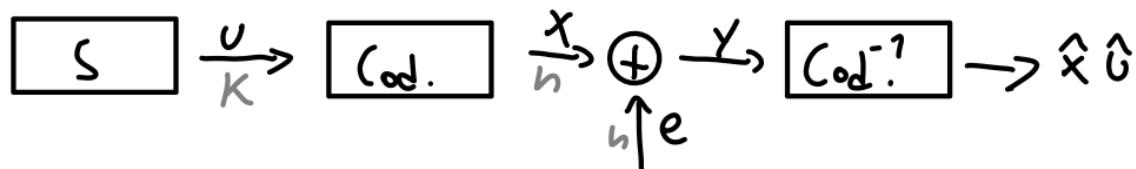
$$p_3 = u_2 + u_3 + u_4$$



- p_1, p_2, p_3 sono i bit di controllo di parità del codice. Si fanno una sorta di somma parziale anzi che totale come il bit di parità singolo che si faceva nel codice a rate $\frac{1}{3}$.
- Per u_1, u_2, u_3 guarda gli 1 nella matrice H che non sono nell'identica

04/04

7 Decodifica codici a blocco



- $\hat{\cdot}$ → indica che il vettore è stimato

Decodifica per i codici a blocco

Dato il vettore ricevuto

$$\mathbf{y} = \mathbf{x} + \mathbf{e}, \quad (1)$$

Il decisore ottimo seleziona la parola di codice $\hat{\mathbf{x}}$ tale che

$$\hat{\mathbf{x}} = \arg \max_{\mathbf{x} \in \mathcal{C}} p(\mathbf{y} | \mathbf{x}) = \arg \min_{\mathbf{x} \in \mathcal{C}} d_H(\mathbf{y}, \mathbf{x}), \quad (2)$$

- ▶ Per ottenere $\hat{\mathbf{x}}$ è necessario fare 2^k confronti fra il vettore ricevuto \mathbf{y} e tutte le parole di codice di $\mathcal{C}(k, n)$;
- ▶ La complessità cresce esponenzialmente con k .

• $\hat{\cdot}$ → stimato

Decodifica per i codici a blocco

Un approccio alternativo consiste nell'osservare che, poiché si ha

$$\mathbf{y} = \mathbf{x} + \mathbf{e} \implies \mathbf{x} = \mathbf{y} + \mathbf{e}, \mathbf{e} = \mathbf{y} - \mathbf{x},$$

la probabilità condizionata può essere riscritta come

$$p(\mathbf{y} | \mathbf{x}) = p(\mathbf{x} + \mathbf{e} | \mathbf{x}) = p(\mathbf{e} | \mathbf{y} + \mathbf{e} \in \mathcal{C}) \quad (3)$$

e quindi, la stima di \mathbf{x} può essere ottenuta come

$$\hat{\mathbf{x}} = \arg \max_{\mathbf{x} \in \mathcal{C}} p(\mathbf{y} | \mathbf{x}) = \mathbf{y} + \arg \max_{\mathbf{e}} p(\mathbf{e} | \{\mathbf{y} + \mathbf{e} \in \mathcal{C}\}) \quad (4)$$

Scrivendo tutti i passaggi vediamo che:

$$\begin{aligned} y &= x + e \rightarrow \\ x &= y - e = y + e \rightarrow \\ e &= y - x \end{aligned}$$

Per quanto riguarda la (3) vediamo perché $p(x+e|x) = p(e|y+e)$: Si puo' eliminare la \mathbf{x} perché è un segnale deterministico(qualcosa di certo) e dunque non influisce sulla probabilità, mentre \mathbf{e} è aleatorio(variabile) come y e quindi non puo' essere eliminato.

Nella (4) vediamo che a y viene sommato il pattern di errore che piu' probabilmente ha corrotto il mio segnale y . Con questo si spera di poter ritrovare x inviato in quanto la somma di una stessa quantità si annulla ovvero:

$$(messaggio) + (\underline{\text{errore}}) + (\underline{\text{errore}}) = (messaggio)$$

Questo accade perche siamo in G(2). Guardare la (6) della prossima slide per la dimostrazione.

Decodifica per i codici a blocco

Invece di stimare direttamente $\hat{\mathbf{x}}$, si stima il vettore errore $\hat{\mathbf{e}}$ più probabile

$$\begin{aligned} \hat{\mathbf{e}} &= \arg \max_{\mathbf{e}} p(\mathbf{e} | \{\mathbf{y} + \mathbf{e} \in \mathcal{C}\}) \\ &= \arg \max_{\mathbf{e} | \{\mathbf{y} + \mathbf{e} \in \mathcal{C}\}} p^{w(\mathbf{e})} (1-p)^{n-w(\mathbf{e})} \\ &= \arg \max_{\mathbf{e} | \{\mathbf{y} + \mathbf{e} \in \mathcal{C}\}} \left(\frac{1-p}{p} \right)^{-w(\mathbf{e})} = \arg \min_{\mathbf{e} | \{\mathbf{y} + \mathbf{e} \in \mathcal{C}\}} w(\mathbf{e}) \end{aligned} \quad (5)$$

- ▶ La decodifica sceglie fra tutti i possibili vettori errore \mathbf{e} tali che $\mathbf{y} + \mathbf{e} \in \mathcal{C}$ quello che ha il peso di Hamming minimo, il minimo numero di errori (*massima verosimiglianza*).
- ▶ Una volta stimato $\hat{\mathbf{e}}$, si ottiene

$$\hat{\mathbf{x}} = \mathbf{y} - \hat{\mathbf{e}} = \mathbf{y} + \hat{\mathbf{e}} = \mathbf{x} + (\mathbf{e} + \hat{\mathbf{e}}) = \begin{cases} \mathbf{x} & \text{if } \hat{\mathbf{e}} = \mathbf{e} \\ \mathbf{x}_1 \neq \mathbf{x} & \text{if } \hat{\mathbf{e}} \neq \mathbf{e} \end{cases}. \quad (6)$$



La produttoria indica la probabilità che cambino i singoli bit del vettore \mathbf{e} . Ricordiamo il concetto di indipendenza trattato qua.

$$\begin{aligned} P(e) &= \prod_{i=1}^n P(e_i) = \\ P(e_i) &= \begin{cases} p & e_i = 1 \\ (1-p) & e_i = 0 \end{cases} \\ P(e) &= p^{w(e)} (1-p)^{n-w(e)} \end{aligned}$$

Il piu' probabile pattern di errori è quello con tutti \emptyset perché se $w(\mathbf{e}) = \emptyset \rightarrow P(e) = (1-p)^n > p^{w(\emptyset)} (1-p)^{n-w(\emptyset)}$

Nella (5) i passaggi completi sono:

$$\begin{aligned} p^{w(e)}(1-p)^{n-w(e)} &= \\ (1-p)^n p^{w(e)}(1-p)^{-w(e)} &= \\ (1-p)^n \left(\frac{p}{1-p}\right)^{w(e)} \end{aligned}$$

{TODO $(1-p)^n$ dove va a finire?} {TODO perché $p < 1-p$?} Siccome $p < 1-p \rightarrow \frac{p}{1-p} < 1$ vogliamo trovare l'esponenziale($w(e)$) piu' piccolo possibile in modo da massimizzarlo. Notare che nella (5) $\left(\frac{p}{1-p}\right)^{w(e)} = \left(\frac{1-p}{p}\right)^{-w(e)}$

7.1 Coset

Decodifica per i codici a blocco

- ▶ *Definizione:* Coset. Sia $\mathcal{C}(k, n)$ un codice a blocco e sia $\mathbf{v} \in \mathcal{V}_n$ un vettore di n cifre binarie, si definisce il *coset* di $\mathcal{C}(k, n)$ individuato da \mathbf{v} l'insieme

$$C_{\mathbf{v}} = C + \mathbf{v} = \{\mathbf{x} + \mathbf{v} : \mathbf{x} \in C\} \quad (7)$$

- ▶ *Proprietà dei coset:*

1. Qualsiasi vettore in \mathcal{V}_n appartiene ad un coset di $\mathcal{C}(k, n)$;
2. Ciascun coset contiene 2^k elementi;
3. Due coset o sono coincidenti o hanno intersezione nulla;
4. Ci sono 2^{n-k} coset distinti;
5. Se \mathbf{v}_1 e \mathbf{v}_2 appartengono allo stesso coset, $\mathbf{v}_1 + \mathbf{v}_2 \in \mathcal{C}(k, n)$ è una parola di codice;

{TODO scrivi dimostrazione 2 e 3}

- Dimostrazione della 2^ proprietà: Potrebbe anche essere che due vettori coincidano, ma questo è

impossibile per le proprietà stesse dei vettori. Dimostriamolo:

HP:

$$v_1, v_2 \in C_v$$

Per assurdo suppongo che:

$$\begin{cases} V_1 = C_1 + V \\ V_2 = C_2 + V \end{cases} \quad \text{e} \quad C_1 \neq C_2$$

Per assurdo dico che: $V_1 = V_2$

ottengo:

$$C_1 + V = C_2 + V$$

$$C_1 = C_2 \rightarrow \text{assurdo}$$

- Dimostrazione della 3^ proprietà:

HP:

$$v \in C_A, C_B \quad C_A = \{V_A + C\} \quad C_B = \{V_B + C\} \quad V_A \notin C_B$$

Si procede con una dimostrazione per assurdo

$$\text{Per assurdo: } V = \begin{cases} C_1 + V_A \in C_A \\ C_2 + V_B \in C_B \end{cases} \quad \text{quindi} \quad C_1 + V_A = C_2 + V_B$$

sommo a sinistra e destra C_1

$$\cancel{C_1} + \cancel{C_1} + V_A = C_2 + C_1 + V_B$$

$$V_A = C + V_B$$

$V_A \in C_B$ ma per ipotesi questo non puo' accadere quindi assurdo

- Dimostrazione della 4^ proprietà:

$$V_1, V_2 \in C_v \Rightarrow \begin{cases} V_1 = C_1 + v \\ V_2 = C_2 + v \end{cases} \Rightarrow v_1 + V_2 = C_1 + \cancel{v} + C_2 + \cancel{v} = \text{altra parola di codice}$$

Questo perché:

$$\begin{cases} C_1 = u_1 G \\ C_2 = u_2 G \end{cases} \Rightarrow C_1 + C_2 = (u_1 + u_2)G$$

{TODO chiedi se coset si puo' vedere come insieme traslato, disegno su xournal}

7.1.1 Esempio di coset

Esempio di coset

Sia $\mathcal{C}(2, 3) = \{000, 101, 010, 111\}$. I coset di $\mathcal{C}(2, 3)$ sono

$$\begin{aligned} C + 000 &= \{000, 101, 010, 111\} = C_0 \\ C + 001 &= \{001, 100, 011, 110\} = C_1 \\ C + 010 &= \{010, 111, 000, 101\} = C_0 \\ C + 011 &= \{011, 110, 001, 100\} = C_1 \\ C + 100 &= \{100, 001, 110, 011\} = C_1 \\ C + 101 &= \{101, 000, 111, 010\} = C_0 \\ C + 110 &= \{110, 011, 100, 001\} = C_1 \\ C + 111 &= \{111, 010, 101, 000\} = C_0 \end{aligned} \tag{8}$$



- Qua si possono vedere alcune proprietà come la 4 infatti abbiamo solo 2 coset poiché:

$$\begin{cases} n = 3 \\ k = 2 \end{cases}$$

e sappiamo che ci sono 2^{n-k} distinti quindi $2^{3-2} = 2$.

7.2 Esempio di decodifica con coset

Decodifica per i codici a blocco

Si può utilizzare il concetto di coset per effettuare la decodifica.

- ▶ Poiché $\mathbf{y} = \mathbf{x} + \mathbf{e}$, dalla definizione di coset discende che i vettori \mathbf{e} e \mathbf{y} appartengono allo stesso coset C_y e che i coset C_y e C_e sono coincidenti.
- ▶ Grazie alle proprietà dei coset, la somma qualsiasi elemento di C_y con \mathbf{y} individua una parola di codice.
- ▶ Il vettore \mathbf{e} va scelto fra gli elementi di C_y e la regola di decisione diventa

$$\hat{\mathbf{e}} = \arg \max_{\mathbf{e}} p(\mathbf{e} | \{\mathbf{y} + \mathbf{e} \in \mathcal{C}\}) = \arg \max_{\mathbf{e} \in C_y} p(\mathbf{e}) = \arg \min_{\mathbf{v} \in C_y} w(\mathbf{v}) \quad (9)$$

- ▶ Tra tutti i 2^k possibili vettori di C_y , il principio di massima verosimiglianza ci dice che devo scegliere quello di peso minimo.



- Dimostriamo cio' che é detto nel primo punto:

1. C_y contiene y che é dato da $x + e$
 2. C_e contiene e
 3. Poiché due coset o sono disgiunti o sono coincidenti e avendo qua sicuramente un punto comune i due sono necessariamente coincidenti.
- Nel secondo punto si fa riferimento alla proprietà (4).

Decodifica per i codici a blocco

Algoritmo di decodifica:

1. Avendo ricevuto il vettore \mathbf{y} , si trova il coset di appartenenza C_y ;
2. Si identifica il *coset leader*, la parola di peso minimo del coset C_y , che è anche la parola di peso minimo del coset C_e ;
3. Il coset leader è la stima del vettore di errore $\hat{\mathbf{e}}$.

Esempio di decodifica utilizzando i coset

Sia $\mathcal{C}(2, 4) = \{0000, 1011, 0101, 1110\}$ la cui $d_{min} = 2$.

I coset sono

$$\begin{aligned}C + 0000 &= \{0000, 1011, 0101, 1110\} \\C + 0001 &= \{0001, 1010, 0100, 1111\} \\C + 0010 &= \{0010, 1001, 0111, 1100\} \\C + 1000 &= \{1000, 0011, 1101, 0110\}\end{aligned}$$

Decodificare i due vettori ricevuti

1. $\mathbf{y} = [1101]$
2. $\mathbf{y} = [1111]$

a) guardo in che coset appartiene: il 4°b) Una volta che ho C_y trovo il coset leader che e' 1000 ovvero quello con meno 1 c) la parola trasmessa e': $\hat{x} = y + 1000 = 0101$

2)

$$y = [1 \ 1 \ 1 \ 1]$$

a) appartiene al 2°coset. b) il leader è 0001 c) la parola trasmessa è $\hat{x} = y + 0001 = 1110$. Però vediamo che qua c'è anche 0100 che ha peso minimo come il leader. Questo problema nasce perché d_{\min} è 2 che è poco e quindi a volte non si riesce a distinguere tra due parlare di codice diverse.

7.3 Decodifica mediante sindrome

La decodifica mediante sindrome è vantaggiosa perché la sindrome ci permette di identificare il coset leader senza dover calcolare tutti i coset singolarmente.

Decodifica mediante sindrome per i codici a blocco

Si definisce *sindrome* di \mathbf{y} , il vettore \mathbf{s} ottenuto dal prodotto di \mathbf{y} con la matrice di controllo di parità

$$\mathbf{s} = \mathbf{y}\mathbf{H}^T = (\mathbf{x} + \mathbf{e})\mathbf{H}^T = \mathbf{x}\mathbf{H}^T + \mathbf{e}\mathbf{H}^T = \mathbf{e}\mathbf{H}^T \quad (10)$$

- ▶ Tutti i membri di uno stesso coset hanno la stessa sindrome;
- ▶ La sindrome \mathbf{s} è composta da $n - k$ cifre binarie;
- ▶ Le 2^{n-k} sindromi sono associate ai 2^{n-k} diversi coset del codice $\mathcal{C}(k, n)$;
- ▶ Ciascuna sindrome è associata ai 2^k pattern di errore appartenenti allo stesso coset.



- $\mathbf{x}\mathbf{H}^T = \emptyset$ per definizione. Vedi 3.7.1
- \mathbf{y} ha n colonne
- \mathbf{H}^T è $n \times (n-k)$
- \mathbf{y} è $1 \times (n-k)$

Decodifica mediante sindrome per i codici a blocco

- ▶ Il decodificatore a sindrome compie quindi le seguenti operazioni:
 1. Calcola la sindrome $\mathbf{s} = \mathbf{y}\mathbf{H}^T$;
 2. Associa la sindrome al coset leader corrispondente $\mathbf{s} \rightarrow \mathbf{e}_{CL}(\mathbf{s})$;
 3. Corregge l'errore sommando il coset leader alla n -upla \mathbf{y}

$$\hat{\mathbf{x}} = \mathbf{y} + \mathbf{e}_{CL}(\mathbf{s}). \quad (11)$$

- ▶ La parola $\hat{\mathbf{x}}$ è una parola di codice:

$$\hat{\mathbf{x}}\mathbf{H}^T = (\mathbf{y} + \mathbf{e}_{CL}(\mathbf{s}))\mathbf{H}^T = \mathbf{s} + \mathbf{s} = 0 \quad (12)$$

- ▶ Per costruzione, la parola di codice $\hat{\mathbf{x}}$ minimizza la distanza di Hamming da \mathbf{y} !



- L'errore $e_{CL}(\mathbf{s})$ è il coset leader del coset associato a S .
- $(y + e_{CL}(s))H^T = s + s$ perché intanto si applica la proprietà distributiva e poi: $yH^T = s$ per la (10) e allo stesso modo $e_{CL}(s)H^T = s$ per la (10). Questo perché e_{CL} è sempre un errore.
- Il terzo punto perché il coset leader ha peso minimo.

Decodifica a sindrome per il codice di Hamming $m = 3$

Il codice ha $d_{min} = 3$ ed è in grado di correggere *esattamente* un errore.

- ▶ Si sceglie la matrice \mathbf{H} in maniera che la *tavella di decodifica* associ alla sindrome il pattern di errore a peso 1 in cui il bit messo a 1 sia nella posizione corrispondente alla conversione della sindrome in decimale.

Codice non sistematico		Codice sistematico	
Syndrome	Coset leader	Syndrome	Coset leader
[000]	[0000000]	[000]	[0000000]
[100]	[1000000]	[100]	[0000100]
[010]	[0100000]	[010]	[0000010]
[110]	[0010000]	[110]	[1000000]
[001]	[0001000]	[001]	[0000001]
[101]	[0000100]	[101]	[0100000]
[011]	[0000010]	[011]	[0010000]
[111]	[0000001]	[111]	[0001000]



- Nella sindrome i numeri sono scritti al contrario quindi il bit meno significativo è a sinistra. E lo stesso vale per come si conta la posizione nel coset leader. Ovvero si parte da sinistra.

I codici di Hamming hanno il vantaggio che la sindrome mi dice qual'è il bit sbagliato. E lo posso quindi correggere. Questo lo faccio moltiplicando la sindrome per un vettore con un 1. Prendiamo un y :

Table 1: questa tabella è						
1	2	3	4	5	6	7
1	0	1	0	1	0	1
0	1	1	0	0	1	1
0	0	0	1	1	1	1

$$[0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0] H^T = [0 \ 0 \ 1]$$

$$y H^T = s(\text{corrispondente}[0 \ 0 \ 1])$$

Se ho un generico y e lo moltiplico per H^T e ottengo $[0 \ 0 \ 1]$ allora sò che l'errore è in posizione 4. Questo però funziona solo se abbiamo solo un errore perché: $2^{n-k} = 2^m = n + 1$.

- $n \rightarrow$ pattern di errore di peso 1.
- $+1 \rightarrow \{\text{TODO}\}$

7.4 Esercizio

Esercizio

- Un codice lineare a blocchi ha la seguente matrice di controllo di parità:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

1. Determinare la matrice generatrice del codice;
2. Decodificare mediante decodifica a sindrome la parola $\mathbf{y} = [110110]$ ed identificare la parola di codice trasmessa.



7.4.1 Svolgimento

1. Notiamo che \mathbf{H} è già in forma sistematica ovvero che a destra abbiamo la matrice identica. Vediamo poi che:

$$\begin{cases} n = 6 \\ n - k = 3 \Rightarrow k = 3 \end{cases} \quad \text{Quindi } \mathbf{G} \text{ è } k \times n$$

Ricordiamo che:

$$\mathbf{H} = [\mathbf{P}^T | \mathbf{I}_{n-k}] \Rightarrow \mathbf{G} = [\mathbf{I}_k | \mathbf{P}]$$

Da questo si deduce che:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

2. $\mathbf{y} = [1 1 0 1 1 0]$

La prima domanda è: È una parola di codice? Per rispondere a questa domanda bisogna verificare che la sindrome abbia tutti \emptyset .

$$s = \mathbf{y}\mathbf{H}^T = [1 1 0 1 1 0] \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [0 1 1]$$

Siccome la sindrome non ha tutti \emptyset ho un errore. Notiamo che H è una base ortogonale per lo spazio vettoriale del codice e questo lo vediamo dal fatto che $GH^T = \emptyset$.

Anziché fare tutti i passaggi illustrati nelle lezioni precedenti come per esempio calcolare i coset posso vedere se la sindrome corrisponde a una riga della matrice H^T (Matrice di controllo di parità). In questo caso vediamo che combacia con la riga numero 2 quindi abbiamo che l'errore è in posizione 2. Per correggere l'errore basta flippare il bit in posizione 2. Vediamo ora analiticamente come si fa.

$$s \rightarrow \hat{e}|y + \hat{e} \in C(k, n) \quad (y + \hat{e})H^T = \emptyset$$

- Questa formula qua sopra vuol dire che la sindrome s mi da l'errore e e quindi sommandolo a y ottengo una parola di codice. Questa parola di codice è il segnale di partenza e infatti se lo moltiplico per H^T mi viene \emptyset .
- \hat{e} sarebbe $e_{CL}(s)$ visto la lezione scorsa

Il vettore che mi seleziona la sindrome è quello che mi identifica l'errore:

$$\begin{aligned} \hat{e} &= [0 \ 1 \ 0 \ 0 \ 0 \ 0]H^T = s \\ &\quad \downarrow \\ \hat{x} &= y + \hat{e} \Rightarrow \hat{x}H^T = (y + \hat{e})H^T = yH^T = s + s = \emptyset \\ y &= [1 \ 1 \ 0 \ 1 \ 1 \ 0] \\ \hat{e} &= [0 \ 1 \ 0 \ 0 \ 0 \ 0] \end{aligned} \Rightarrow \hat{x} = [1 \ 0 \ 0 \ 1 \ 1 \ 0]$$

Facciamo ora la verifica:

$$\hat{x}H^T = [0 \ 0 \ 0]$$

18/04

7.5 Codici ciclici

I Codici ciclici sono codici lineari.

Codici ciclici - Definizione

- ▶ Dato il vettore $\mathbf{v} = [v_0, v_1, \dots, v_{n-1}] \in \mathcal{V} = \text{GF}(2)^n$ indichiamo con $\mathbf{v}^{(i)}$ il vettore ottenuto da \mathbf{v} applicando uno shift ciclico di i posizioni a destra

$$\mathbf{v}^{(i)} = [v_{n-i}, v_{n-i+1}, \dots, v_{n-1}, v_0, v_1, \dots, v_{n-i-1}]$$

- ▶ Un codice lineare $\mathcal{C}(k, n)$ si definisce *ciclico* se data una generica parola di codice $\mathbf{x} \in \mathcal{C}(k, n)$ ogni suo shift ciclico $\mathbf{x}^{(i)}$ è ancora una parola di codice, $\mathbf{x}^{(i)} \in \mathcal{C}(k, n)$.

7.5.1 Esempi

Codici ciclici - Esempi

- ▶ $\mathcal{C}(2, 3) = \{000, 110, 101, 011\}$
- ▶ $\mathcal{C}(2, 4) = \{0000, 1010, 0101, 1111\}$
- ▶ Il codice $\mathcal{C}(4, 7)$

Messages	Code Vectors	Code Polynomials
(0 0 0 0)	0 0 0 0 0 0 0	$0 = 0 \cdot \mathbf{g}(x)$
(1 0 0 0)	1 1 0 1 0 0 0	$1 + x + x^3 = 1 \cdot \mathbf{g}(x)$
(0 1 0 0)	0 1 1 0 1 0 0	$x + x^2 + x^4 = x \cdot \mathbf{g}(x)$
(1 1 0 0)	1 0 1 1 1 0 0	$1 + x^2 + x^3 + x^4 = (1 + x) \cdot \mathbf{g}(x)$
(0 0 1 0)	0 0 1 1 0 1 0	$x^2 + x^3 + x^5 = x^2 \cdot \mathbf{g}(x)$
(1 0 1 0)	1 1 1 0 0 1 0	$1 + x + x^2 + x^5 = (1 + x^2) \cdot \mathbf{g}(x)$
(0 1 1 0)	0 1 0 1 1 1 0	$x + x^3 + x^4 + x^5 = (x + x^2) \cdot \mathbf{g}(x)$
(1 1 1 0)	1 0 0 0 1 1 0	$1 + x^4 + x^5 = (1 + x + x^2) \cdot \mathbf{g}(x)$
(0 0 0 1)	0 0 0 1 1 0 1	$x^3 + x^4 + x^6 = x^3 \cdot \mathbf{g}(x)$
(1 0 0 1)	1 1 0 0 1 0 1	$1 + x + x^4 + x^6 = (1 + x^3) \cdot \mathbf{g}(x)$
(0 1 0 1)	0 1 1 1 0 0 1	$x + x^2 + x^3 + x^6 = (x + x^3) \cdot \mathbf{g}(x)$
(1 1 0 1)	1 0 1 0 0 0 1	$1 + x^2 + x^6 = (1 + x + x^3) \cdot \mathbf{g}(x)$
(0 0 1 1)	0 0 1 0 1 1 1	$x^2 + x^4 + x^5 + x^6 = (x^2 + x^3) \cdot \mathbf{g}(x)$
(1 0 1 1)	1 1 1 1 1 1 1	$1 + x + x^2 + x^3 + x^4 + x^5 + x^6 = (1 + x^2 + x^3) \cdot \mathbf{g}(x)$
(1 1 1 1)	1 0 0 1 0 1 1	$1 + x^3 + x^5 + x^6 = (1 + x + x^2 + x^3) \cdot \mathbf{g}(x)$

7.5.2 Rappresentazione algebrica

Rappresentazione algebrica di un codice ciclico

- ▶ A ciascun vettore $\mathbf{v} = [v_0, v_1, \dots, v_{n-1}] \in \mathcal{V}$ è possibile associare un polinomio definito in $\text{GF}(2)$,

$$v(D) = v_0 + v_1D + \dots + v_{n-1}D^{n-1},$$

$$\mathbf{v} \leftrightarrow v(D).$$

- ▶ *Definizione.* Se $x(D) \leftrightarrow \mathbf{x} \in \mathcal{C}(k, n) \implies$ si dice che $x(D)$ è in $\mathcal{C}(k, n)$.
- ▶ *Proprietà.* Uno shift ciclico di i posizioni del vettore \mathbf{v} è equivalente a moltiplicare il polinomio $v(D)$ per D^i modulo $(D^n - 1)$

$$\mathbf{v}^{(i)} \leftrightarrow \mod \{D^i v(D), (D^n - 1)\}.$$

Rappresentazione algebrica di un codice ciclico

Ad esempio, fissando $i = 1$, il polinomio diventa

$$\begin{aligned}Dv(D) &= v_0D + v_1D^2 + \cdots + v_{n-1}D^n \\&= v_{n-1} + v_0D + v_1D^2 + \cdots + v_{n-2}D^{n-1} + v_{n-1}D^n - v_{n-1} \\&= v_{n-1} + v_0D + v_1D^2 + \cdots + v_{n-2}D^{n-1} + v_{n-1}(D^n - 1)\end{aligned}$$

In questo caso si ha

$$\text{mod } \{Dv(D), (D^n - 1)\} = v_{n-1} + v_0D + v_1D^2 + \cdots + v_{n-2}D^{n-1} \leftrightarrow \mathbf{v}^{(1)}$$

Analogamente, si può mostrare che

$$D^i v(D) = q(D)(D^n - 1) + v_{n-i} + v_{n-i+1}D + \cdots + v_{n-i-1}D^{n-1},$$

e quindi

$$\text{mod } \{D^i v(D), (D^n - 1)\} \leftrightarrow \mathbf{v}^{(i)}$$

Rappresentazione algebrica di un codice ciclico

Terorema. Sia $g(D) = g_0 + g_1D + \cdots + g_rD^r$ il polinomio di grado minimo associato ad una parola del codice ciclico $\mathcal{C}(k, n)$ \implies a) $g_0 = 1$, e b) $g(D)$ è unico.

Dimostrazione.

- a) Supponiamo *per assurdo* che $g_0 = 0 \implies$

$$\begin{aligned} g(D) &= g_1D + g_2D^2 + \cdots + g_rD^r \\ &= D(g_1 + g_2D + \cdots + g_rD^{r-1}) = Dg'(D) \end{aligned}$$

ma questo contraddice le ipotesi poiché $\mathcal{C}(k, n)$ è ciclico e quindi $g'(D)$ è in $\mathcal{C}(k, n)$ ma il grado di $g'(D)$ è minore di r . Un ragionamento simile si può fare per $g_r = 0$.

- b) Supponiamo che esistano *due* polinomi di grado minimo $g_1(D)$ e $g_2(D)$ in $\mathcal{C}(k, n) \implies g_3(D) = g_1(D) - g_2(D)$ è ancora in $\mathcal{C}(k, n)$ e per quanto visto nella parte a) il grado di $g_3(D)$ sarebbe minore di r . Impossibile.

7.5.3 Polinomio generatore

Polinomio generatore di un codice ciclico

Definizione. Il *polinomio generatore* di un codice ciclico $\mathcal{C}(k, n)$ è il polinomio

$$g(D) = 1 + g_1 D + \cdots + D^r,$$

non nullo e di grado minimo in $\mathcal{C}(k, n)$.

7.5.4 Esempi

Codici ciclici - Esempi

- ▶ $\mathcal{C}(2, 3) = \{000, 110, 101, 011\} \implies g(D) = 1 + D$
- ▶ $\mathcal{C}(2, 4) = \{0000, 1010, 0101, 1111\} \implies g(D) = 1 + D^2$
- ▶ Il codice $\mathcal{C}(4, 7) \implies g(D) = 1 + D + D^3$

Messages	Code Vectors	Code Polynomials
(0 0 0 0)	0 0 0 0 0 0 0	$0 = 0 \cdot \mathbf{g}(x)$
(1 0 0 0)	1 1 0 1 0 0 0	$1 + x + x^3 = 1 \cdot \mathbf{g}(x)$
(0 1 0 0)	0 1 1 0 1 0 0	$x + x^2 + x^4 = x \cdot \mathbf{g}(x)$
(1 1 0 0)	1 0 1 1 1 0 0	$1 + x^2 + x^3 + x^4 = (1 + x) \cdot \mathbf{g}(x)$
(0 0 1 0)	0 0 1 1 0 1 0	$x^2 + x^3 + x^5 = x^2 \cdot \mathbf{g}(x)$
(1 0 1 0)	1 1 1 0 0 1 0	$1 + x + x^2 + x^5 = (1 + x^2) \cdot \mathbf{g}(x)$
(0 1 1 0)	0 1 0 1 1 1 0	$x + x^3 + x^4 + x^5 = (x + x^2) \cdot \mathbf{g}(x)$
(1 1 1 0)	1 0 0 0 1 1 0	$1 + x^4 + x^5 = (1 + x + x^2) \cdot \mathbf{g}(x)$
(0 0 0 1)	0 0 0 1 1 0 1	$x^3 + x^4 + x^6 = x^3 \cdot \mathbf{g}(x)$
(1 0 0 1)	1 1 0 0 1 0 1	$1 + x + x^4 + x^6 = (1 + x^3) \cdot \mathbf{g}(x)$
(0 1 0 1)	0 1 1 1 0 0 1	$x + x^2 + x^3 + x^6 = (x + x^3) \cdot \mathbf{g}(x)$
(1 0 1 0)	1 0 1 0 0 0 1	$1 + x^2 + x^6 = (1 + x + x^3) \cdot \mathbf{g}(x)$
(0 0 1 1)	0 0 1 0 1 1 1	$x^2 + x^4 + x^5 + x^6 = (x^2 + x^3) \cdot \mathbf{g}(x)$
(1 0 1 1)	1 1 1 1 1 1 1	$1 + x + x^2 + x^3 + x^4 + x^5 + x^6 = (1 + x^2 + x^3) \cdot \mathbf{g}(x)$
(1 1 1 1)	1 0 0 1 0 1 1	$1 + x^3 + x^5 + x^6 = (1 + x + x^2 + x^3) \cdot \mathbf{g}(x)$

Per capire come viene ricavato $g(D)$ che e' il polinomio generatore (di grado minimo) facciamo un esempio con $\mathcal{C}(2,3)$ vediamo che il minore e' 110 perche':

$$1D^0 + 1D^1 + 0D^2 = 1 + D$$

7.5.5 Polinomio generatore parte-2

Polinomio generatore di un codice ciclico

Teorema. Un polinomio $x(D)$ è in $\mathcal{C}(k, n) \iff x(D)$ è un multiplo di $g(D)$.

Dimostrazione.

- a) Ogni multiplo di $g(D)$ è in $\mathcal{C}(k, n)$. Poiché $\mathcal{C}(k, n)$ è ciclico i polinomi $Dg(D), D^2g(D), \dots, D^{n-r-1}g(D)$ sono in $\mathcal{C}(k, n)$ e lo è anche qualsiasi loro combinazione lineare $x(D) = u(D)g(D) = u_0g(D) + u_1Dg(D) + \dots + u_{n-r-1}D^{n-r-1}g(D)$
- b) Ogni polinomio in $\mathcal{C}(k, n)$ può essere espresso come multiplo di $g(D)$. Per assurdo assumiamo che $x(D)$ sia in $\mathcal{C}(k, n)$ ma non un multiplo di $g(D) \implies x(D) = a(D)g(D) + b(D)$, $b(D) = x(D) - a(D)g(D)$. Poichè sia $x(D)$ che $a(D)g(D)$ sono in $\mathcal{C}(k, n)$, per la linearità del codice anche $b(D)$ è in $\mathcal{C}(k, n)$ ma questo è impossibile perché $b(D)$, essendo il resto alla divisione di $x(D)$ per $g(D)$, è di grado minore di $g(D)$.

Polinomio generatore di un codice ciclico

Corollario 1. L'insieme degli $n - r$ polinomi

$$\{g(D), Dg(D), D^2g(D), \dots, D^{n-r-1}g(D)\}$$

costituisce una base per $\mathcal{C}(k, n)$.

Corollario 2. Se il polinomio generatore $g(D)$ del codice $\mathcal{C}(k, n)$ ha grado $r \implies$ il numero di parole del codice è 2^{n-r} e $r = n - k$.

Dimostrazione. Tutte le possibili combinazioni in GF(2) degli $n - r$ polinomi che costituiscono una base per $\mathcal{C}(k, n)$ sono 2^{n-r} e quindi le parole di codice sono $2^{n-r} \implies k = n - r$.

Corollario 3. Il grado del polinomio generatore $g(D)$ del codice $\mathcal{C}(k, n)$ è uguale al numero di bit di controllo di parità.

7.5.6 Teorema fondamentale generatore di un codice ciclico

Teorema fondamentale generatore di un codice ciclico

Teorema. Un polinomio $g(D)$ è generatore di un codice ciclico $\mathcal{C}(k, n) \iff g(D)$ è un divisore di $D^n - 1$.

Dimostrazione.

- a) Il polinomio $g(D)$ è generatore di $\mathcal{C}(k, n) \implies g(D)$ è un divisore di $D^n - 1$.

Poiché $g(D)$ è di grado $n - k$, si ha che

$$D^k g(D) = (D^n - 1) + g^{(k)}(D),$$

da cui

$$(D^n - 1) = D^k g(D) - g^{(k)}(D) = \left(D^k - a(D) \right) g(D)$$

Teorema fondamentale generatore di un codice ciclico

- b) Se il polinomio $g(D)$ di grado $n - k$ è un divisore di $D^n - 1 \implies g(D)$ è generatore di un codice ciclico $\mathcal{C}(k, n)$. Qualsiasi polinomio della forma

$$x(D) = u_0g(D) + u_1Dg(D) + \cdots + u_{k-1}D^{k-1}g(D) = u(D)g(D)$$

ha grado pari o inferiore a $n - 1$ ed è un multiplo di $g(D)$.

Poichè $u(D)$ può assumere 2^k valori \implies l'insieme dei 2^k vettori forma un codice lineare $\mathcal{C}(k, n)$.

Sia $v(D) = v_0 + v_1D + \cdots + v_{n-1}D^{n-1} = a(D)g(D)$ in $\mathcal{C}(k, n) \implies$

$$\begin{aligned} Dv(D) &= v_0D + v_1D^2 + \cdots + v_{n-1}D^n \\ &= v_{n-1}(D^n - 1) + (v_{n-1} + v_0D + \cdots + v_{n-2}D^{n-1}) \\ &= v_{n-1}(D^n - 1) + v^{(1)}(D) \end{aligned}$$

Poichè $g(D)$ è divisore di $D^n - 1$ (per ipotesi) e di

$Dv(D) = Da(D)g(D)$ anche $v^{(1)}(D)$ è un multiplo di $g(D)$ $\implies \mathcal{C}(k, n)$ è ciclico.

7.5.7 Divisione tra polinomi in GF(2)

Divisione tra polinomi in GF(2) - Esempi

► $\mathcal{C}(2, 3) = \{000, 110, 101, 011\} \implies g(D) = 1 + D.$

$$(D^3 - 1)/(1 + D) = D^2 + D + 1.$$

► $\mathcal{C}(2, 4) = \{0000, 1010, 0101, 1111\} \implies g(D) = 1 + D^2$

$$(D^4 - 1)/(1 + D^2) = D^2 + 1.$$

► Il codice $\mathcal{C}(4, 7) \implies g(D) = 1 + D + D^3$

$$(D^7 - 1)/(1 + D + D^3) = D^4 + D^2 + D + 1.$$

► Il polinomio $D^6 - 1$ può essere fattorizzato in molte maniere diverse. Ad ogni fattore corrisponde un polinomio generatore $g(D)$ diverso e quindi un codice cirlico diverso.

$$(D^6 - 1) = (1 + D^2)^2(1 + D + D^2)^2.$$

Esempi di divisioni svolte

D^3	0	0	1	D	+1
<hr/>					
D^3	D^2				
<hr/>					
D^2	0	1			
<hr/>					
D^2	D				
<hr/>					
D	1				
<hr/>					
D	1				
<hr/>					
			0		

$$\begin{array}{cccccccc|ccc}
 D^7 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & D^3 & D & 1 \\
 D^7 & 0 & D^5 & D^4 & & & & & & & \\
 \hline
 & D^5 & D^4 & & & & & & & & \\
 D^5 & 0 & D^3 & D^2 & & & & & & & \\
 \hline
 & D^4 & D^3 & D^2 & & & 1 & & & & \\
 D^4 & 0 & D^2 & D & & & & & & & \\
 \hline
 & D^3 & 0 & D & 1 & & & & & & \\
 D^3 & 0 & D & 1 & & & & & & & \\
 \hline
 & & & & & & 0 & & & &
 \end{array}$$

7.5.8 Matrice generatrice di un codice ciclico

Matrice generatrice di un codice ciclico

Dato il codice ciclico $\mathcal{C}(k, n)$ con polinomio generatore $g(D)$, dal momento che l'insieme dei polinomi

$$\{g(D), Dg(D), D^2g(D), \dots, D^{k-1}g(D)\}$$

costituisce una base per il codice, la matrice generatrice del codice è

$$\mathbf{G} = \begin{bmatrix} g_0 & g_1 & \dots & g_{n-k} & 0 & \dots & 0 \\ 0 & g_0 & g_1 & \dots & g_{n-k} & \dots & 0 \\ \vdots & & & & & & \vdots \\ 0 & \dots & 0 & g_0 & g_1 & \dots & g_{n-k} \end{bmatrix}$$

Considerato che $g_0 = 1$, le righe possono essere sommate tra loro per ottenere la matrice generatrice del codice equivalente in forma sistematica.



- La matrice G è composta dalla base shiftata di 1 per ogni riga.
- x è di grado n-1 e h è di grado k e v che è il prodotto dei due è n+k-1

7.5.9 Controllo di parità per un codice ciclico

Controllo di parità per un codice ciclico

Dato il codice ciclico $\mathcal{C}(k, n)$ con polinomio generatore $g(D)$, esiste sempre un polinomio $h(D) = h_0 + h_1D + \cdots + h(k)D^k$ tale che

$$h(D) = (D^n - 1) / g(D) \implies g(D)h(D) = D^n - 1.$$

- ▶ Sia $x(D) = u(D)g(D)$ in $\mathcal{C}(k, n) \implies$

$$\begin{aligned} v(D) &= x(D)h(D) = u(D)g(D)h(D) \\ &= u(D)(D^n - 1) = D^n u(D) - u(D). \end{aligned}$$

- ▶ Poiché $u(D)$ è un polinomio di grado massimo $k - 1$ e $D^n u(D)$ è di grado minimo n , sappiamo per certo che, se $x(D)$ è in $\mathcal{C}(k, n)$, gli $n - k$ coefficienti con indici $k, k + 1, \dots, n - 1$ del polinomio $v(D)$ devono essere 0.



$$D^n u(D) = D^n(u_0 + u_1D + \cdots + u_{k-1}D^{k-1})$$

- Il grado minimo di questo polinomio è n
- Tra n e k-1 abbiamo tutti \emptyset

I coefficienti tra k e n-1 sono n-k che è la dimensione della matrice di controllo di parità

Controllo di parità per un codice ciclico

- Poiché è $v(D) = x(D)h(D)$, il coefficiente m -esimo del polinomio $v(D)$ si ottiene come la somma di tutti i coefficienti che moltiplicano D^m

$$v_m = x_0 h_m + x_1 h_{m-1} + \cdots + x_m h_0 = \sum_{j=0}^{n-1} x(j)h(m-j).$$

- Abbiamo un set di $n - k$ equazioni del tipo

$$v_m = \sum_{j=0}^{n-1} x(j)h(m-j) = 0, \quad m = k, k+1, \dots, n-1.$$

- x_0 è di grado 0, x_1 di grado 1, etc. Se lo moltiplicherà per gli h il grado finale di ogni elemento della somma è m .
- In pratica ti sta dicendo come trovare la somma dei coefficienti del prodotto di due polinomi che hanno grado m .

Controllo di parità per un codice ciclico

- Le $n - k$ equazioni possono essere riassunte in forma matriciale

$$\mathbf{x}\mathbf{H}^T = \mathbf{0}_{n-k}.$$

dove la matrice \mathbf{H} , di dimensioni $(n - k) \times n$, è la matrice di controllo di parità del codice ciclico $\mathcal{C}(k, n)$.

$$\mathbf{H}^T = \begin{bmatrix} h_k & h_{k-1} & \dots & h_0 & 0 & \dots & 0 \\ 0 & h_k & h_{k-1} & \dots & h_0 & \dots & 0 \\ \vdots & & & & & & \vdots \\ 0 & \dots & 0 & h_k & h_{k-1} & \dots & h_0 \end{bmatrix}$$

- La sindrome può essere calcolata come

$$\mathbf{s} = \mathbf{y}\mathbf{H}^T$$

- Con matrici h diverse cambia la sindrome

7.5.10 Esempio di calcolo di matrice generatrice e di controllo di parità

Esempio di calcolo di matrice generatrice e di controllo di parità

Dato il codice ciclico $\mathcal{C}(k = 4, n = 7)$ con polinomio generatore $g(D) = 1 + D + D^3$, la matrice generatrice è

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

ed in forma sistematica diventa

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Esempio di calcolo di matrice generatrice e di controllo di parità

Dato il codice ciclico $\mathcal{C}(k = 4, n = 7)$ con polinomio generatore $g(D) = 1 + D + D^3$, il vettore $h(D) = (D^n - 1)/g(D) = 1 + D + D^2 + D^4$. I coefficienti del polinomio sono $h_0 = 1, h_1 = 1, h_2 = 1, h_3 = 0, h_4 = 1$ e la matrice di controllo di parità è

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

ed in forma sistematica diventa

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

7.5.11 Metodo alternativo per calcolo della sindrome

Metodo alternativo per il calcolo della sindrome

La sindrome associata alla matrice di controllo di parità sistematica si può calcolare usando un metodo alternativo.

Al ricevitore si ha

$$\mathbf{y} = \mathbf{x} + \mathbf{e} \implies y(D) = x(D) + e(D)$$

La sindrome di \mathbf{y} può essere calcolata come il resto della divisione tra polinomi $y(D)/g(D) \implies y(D) = a(D)g(D) + s(D)$.



- $a(D) \rightarrow$ quoziente
- $s(D) \rightarrow$ resto

$$mod\left(\frac{y(D)}{g(D)}\right) = mod\left(\frac{x(D) + e(D)}{g(D)}\right) = mod\left(\frac{\cancel{x(D)}}{\cancel{g(D)}} + \frac{e(D)}{g(D)}\right)$$

- Il grado di $g(D)$ è $(n-k) \rightarrow (n-k)$ coefficienti perchè c'è lo \emptyset

Se il grado di $e(D)$ è minore del grado di $g(D)$ abbiamo che:

$$mod\{e(D), g(D)\} = e(D)$$

- $e(D)$ è la sindrome

$$e(D) = u(D)g(D) + s(D)$$

Metodo alternativo per il calcolo della sindrome

Poiché il grado di $s(D)$ è minore del grado di $g(D)$, il grado massimo di $s(D)$ è $n - k - 1$.

- ▶ Se è $e(D) = 0$, il canale non introduce errori ed è

$$s(D) = \text{mod} \{x(D), g(D)\} = \text{mod} \{u(D)g(D), g(D)\} = 0$$

- ▶ Se è $e(D) \neq 0$, la sindrome è

$$\begin{aligned} s(D) &= \text{mod} \{x(D) + e(D), g(D)\} \\ &= \text{mod} \{x(D), g(D)\} + \text{mod} \{e(D), g(D)\} \\ &= \text{mod} \{e(D), g(D)\} \end{aligned}$$

- ▶ $s(D)$ corrisponde alla sindrome ottenuta con la matrice di controllo di parità in forma *sistematica*.

7.5.12 Decodifica a sindrome per codici ciclici

Decodifica a sindrome per codici ciclici

Ci sono due metodi possibili per effettuare la decodifica dei codici ciclici:

1. Approccio classico codici a blocco: La sindrome $s(D)$ mappa un polinomio di grado $n - 1$ su uno di grado $n - k - 1$. Una volta calcolata la sindrome, si identifica un coset ed il pattern di errore corrisponde al coset leader. *Svantaggio*: la complessità di associare tutti i vettori in \mathcal{V} ad uno specifico coset.
2. Sfruttare le proprietà dei codici ciclici per derivare un metodo alternativo.

Decodifica a sindrome per codici ciclici

Teorema. Dato il codice cirlico $\mathcal{C}(k, n)$ con polinomio generatore $g(D)$ e distanza minima d_{min} , sia $s(D)$ la sindrome associata al vettore ricevuto \mathbf{y} , se $w(s(D)) \leq \lfloor \frac{d_{min}-1}{2} \rfloor \implies \hat{e}(D) = s(D)$.

Dimostrazione. Per costruzione $s(D)$ e $y(D)$ sono nello stesso coset $C_y = C_s = \{\mathcal{C} + s(D)\}$, poiché si ha $w(\mathbf{s}) \leq \lfloor \frac{d_{min}-1}{2} \rfloor \implies s(D) \leftrightarrow [\mathbf{s}, 0 \dots, 0]$ è il coset leader e quindi la stima dell'errore.

In altre parole, poiché $s(D) = \text{mod}\{e(D), g(D)\}$, se il grado di $e(D) < n - k \implies s(D) = e(D)$.



- $s(D)$ e $y(D)$ sono nello stesso coset perché resto della divisione tra $e(D)$ e $g(D)$ è quindi sempre nello stesso coset.

7.5.13 Esempio di decodifica per codici ciclici

Esempio di decodifica per codici ciclici

Sia $\mathbf{x} = [0110100]$ una parola del codice ciclico $\mathcal{C}(4, 7)$ con polinomio generatore $g(D) = 1 + D + D^3$.

- ▶ Sia $\mathbf{e} = [0100000]$ l'errore introdotto dal canale. Il vettore ricevuto è $\mathbf{y} = \mathbf{x} + \mathbf{e} = [0010100] \leftrightarrow y(D) = D^2 + D^4$.

$$s(D) = \text{mod}\{y(D), g(D)\} = D$$

Poichè $w(s(D)) = 1 \implies s(D) = \hat{e}(D) \implies \hat{\mathbf{e}} = [0100000]$.

- ▶ Sia $\mathbf{e} = [0000010]$ l'errore introdotto dal canale. Il vettore ricevuto è
 $\mathbf{y} = \mathbf{x} + \mathbf{e} = [0110110] \leftrightarrow y(D) = D + D^2 + D^4 + D^5$.

$$s(D) = \text{mod}\{y(D), g(D)\} = D^2 + D + 1$$

Poichè $w(s(D)) = 3 \implies s(D) \neq \hat{e}(D)$ e per trovare l'errore bisogna trovare un altro metodo.



Decodifica a sindrome per codici ciclici

Teorema. Dato il codice ciclico $\mathcal{C}(k, n)$, sia $s(D)$ la sindrome del vettore ricevuto $\mathbf{y} \implies$ la sindrome $s_1(D)$ della parola $\mathbf{y}^{(1)}$ ottenuta dallo shift di ciclico di \mathbf{y} di una posizione si calcola

$$s_1(D) = \text{mod}\left\{y^{(1)}(D), g(D)\right\} = Ds(D) - s_{n-k-1}g(D)$$

Dimostrazione. Poiché vale la relazione $y(D) = u(D)g(D) + s(D)$, la relazione relativa a $y^{(1)}(D)$ è

$$\begin{aligned} Dy(D) &= Du(D)g(D) + Ds(D) \\ &= (Du(D)g(D) + s_{n-k-1})g(D) + Ds(D) - s_{n-k-1}g(D) \end{aligned}$$

Poiché il grado massimo di $Ds(D) - s_{n-k-1}g(D)$ è $n - k - 1 \implies s_1(D) = Ds(D) - s_{n-k-1}g(D)$ è il resto della divisione di $Dy(D)$ per $g(D)$ ed è quindi è la sindrome di $y^{(1)}(D)$.



- S_{n-k-1} o è \emptyset o è 1
- Nel calcolo di $D_y(D)$ sommo e sottraggo $S_{n-k-1}g(D)$
- Nel calcolo di $D_y(D)$ la parentesi è stata sbagliata dal prof. Non ci va $g(D)$ perché lo ho appena raccolto

Esempio di decodifica per codici ciclici

Sia $\mathbf{x} = [0110100]$ una parola del codice cirlico $\mathcal{C}(4, 7)$ con polinomio generatore $g(D) = 1 + D + D^3$ ed $\mathbf{e} = [0000010]$ l'errore introdotto dal canale. Il vettore ricevuto è $\mathbf{y} = \mathbf{x} + \mathbf{e} = [0110110] \leftrightarrow y(D) = D + D^2 + D^4 + D^5$.

$$s(D) = \text{mod}\{y(D), g(D)\} = D^2 + D + 1.$$

Lo shift ciclico di \mathbf{y} è $\mathbf{y}^{(1)} = [0011011] \leftrightarrow y^{(1)}(D) = D^2 + D^3 + D^5 + D^6$.

$$s_1(D) = \text{mod}\{y^{(1)}(D), g(D)\} = D^2 + 1 = D(D^2 + D + 1) - D^3 + D + 1.$$

Lo shift ciclico di $\mathbf{y}^{(1)}$ è $\mathbf{y}^{(2)} = [1001101] \leftrightarrow y^{(2)}(D) = 1 + D^3 + D^4 + D^6$.

$$s_2(D) = \text{mod}\{y^{(2)}(D), g(D)\} = 1 = D(D^2 + 1) - D^3 + D + 1.$$



- Ricordarsi il teorema che dice che il peso della sindrome ovvero il numero di 1 presenti nel suo vettore associato deve essere minore di d_{\min} . Guarda il teorema. Siccome stiamo considerando codici di hamming la $d_{\min} = 3$.
- $S_1(D)$ = Sindrome dello shift di 1 posizione
- $D_s(D)$ = shift ciclico

$$\begin{aligned} y(D) &= D + D^2 + D^4 + D^5 \\ e(D) &= D^5 \end{aligned}$$

D^5	D^4	0	D^2	D	0	D^3	D	1
D^5	0	D^3	D^2	0	0	D^2	D	1
D^4	D^3	0	D	0				
D^4	0	D^2	D	0				
D^3	D^2	0	0					
D^3	0	D	1					
D^2	D	1						

$$s(D) = D^2 + D + 1$$

$w(s(D)) = 3$ perché il vettore s ha 3 elementi

Andiamo ora con un altro esempio:

$$y = [0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0]$$

$$y^{(1)} = [0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1] \leftrightarrow y^{(1)}(D) = D^2 + D^3 + D^5 + D^6$$

$S_1(D)$ calcolata col metodo tradizionale =

$$\begin{array}{ccccccc|ccc}
 D^6 & D^5 & 0 & D^3 & D^2 & 0 & 0 & D^3 & D & 1 \\
 \hline
 D^6 & 0 & D^4 & D^3 & & & & D^3 & D^2 & D & 1 \\
 \hline
 D^5 & D^4 & 0 & D^2 & & & & & & & \\
 \hline
 D^5 & 0 & D^3 & D^2 & & & & & & & \\
 \hline
 D^4 & D^3 & 0 & & & & & & & & \\
 \hline
 D^4 & 0 & D^2 & D & & & & & & & \\
 \hline
 D^3 & D^2 & 0 & 0 & & & & & & & \\
 \hline
 D^3 & & D & 1 & & & & & & & \\
 \hline
 D^2 & 0 & 1 & & & & & & & & \\
 \end{array}$$

Il metodo che ci suggerisce il teorema è:

$$\begin{aligned}
 s_1(D) &= D_s(D) - s_2 g(D) = D(D^2 + D + 1) - D^3 + D + 1 = \\
 &D^3 + D^2 - D^3 - D - 1 = D^2 - 1 = \\
 &D^2 + 1
 \end{aligned}$$

È molto più veloce e si ottiene ovviamente lo stesso risultato

Decodifica a sindrome per codici ciclici

Definizione. Dato un vettore \mathbf{v} di n componenti, una *sequenza ciclica* di zeri di lunghezza ℓ è una successione di ℓ zeri consecutivi in senso ciclico.

Esempi

1. $n = 7, \ell = 3, \mathbf{v} = [1000101];$
2. $n = 7, \ell = 4, \mathbf{v} = [0010100];$
3. $n = 15, \ell = 9, \mathbf{v} = [000000110001000]$

Decodifica a sindrome per codici ciclici

Teorema. Dato il codice cirlico $\mathcal{C}(k, n)$, con polinomio generatore $g(D)$ e distanza minima d_{min} , tale che tutti i pattern di errore correggibili abbiano una *sequenza ciclica* di almeno k zeri, ricevuto il vettore $\mathbf{y} = \mathbf{x} + \mathbf{e}$ con $w(\mathbf{e}) \leq \lfloor \frac{d_{min}-1}{2} \rfloor$, l'algoritmo di decodifica a massima verosimiglianza è composto dai seguenti passi

1. Calcolo iterativamente le sindromi di $s_i(D)$ per tutti gli shift cirlici di $y(D)$ e computo $w(s_i(D))$;
2. Trovo m per cui $w(s_m(D)) \leq \lfloor \frac{d_{min}-1}{2} \rfloor$;
3. Stimo $\hat{\mathbf{e}}(D) = \text{mod} \{ D^{n-m} s_m(D), D^{n-1} \}$.

- $w(\mathbf{e})$ indica il limite per cui l'errore sia correggibile

- Nella 3 il modulo è il controshift della sindrome

Decodifica a sindrome per codici ciclici

Dimostrazione.

1. *Esistenza di m.* Poichè $w(\mathbf{e}) \leq \lfloor \frac{d_{min}-1}{2} \rfloor$ e tutti i pattern di errore correggibili hanno una sequenza ciclica di almeno k zeri, esiste uno shift ciclico di m posizioni di \mathbf{y} tale che tutti gli 1 di \mathbf{e} siano compresi nelle prime $n - k$ posizioni di $\mathbf{y}^m \implies s_m(D) = e^{(m)}(D)$.
2. *Stima dell'errore*

$$\begin{aligned}
 D^m (y(D) + D^{n-m} s_m(D)) &= D^m y(D) + D^n s_m(D) \\
 &= y^{(m)}(D) + D^n s_m(D) \\
 &= u(D)g(D) + s_m(D) + D^n s_m(D) \\
 &= u(D)g(D) + (D^n - 1)s_m(D) \\
 &= (u(D) + h(D)s_m(D))g(D).
 \end{aligned}$$

8 Codici convoluzionali

8.1 Introduzione

Introduzione ai codici convoluzionali

Poiché sono molto potenti in termini di capacità di correzione di errore e, dall'introduzione dell'Algoritmo di Viterbi, hanno complessità limitata, i codici convoluzionali sono una famiglia di codici lineari utilizzati in moltissime applicazioni, tra cui:

- ▶ WiFi (802.11) e reti cellulari 2G, 3G, 4G e 5G.
- ▶ Tv digitale (DVB-T).
- ▶ Deep space communications.



[Credit: Nasa Ames Research Center]

Introduzione ai codici convoluzionali

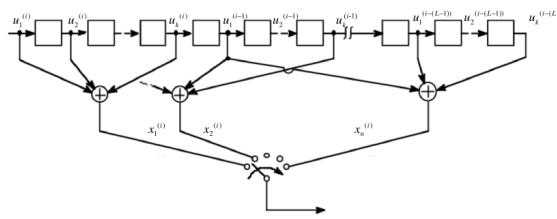
- ▶ Diversamente dai codici a blocco in forma sistematica, i *codici convoluzionali* sono in generale non-sistematici.
- ▶ In un codice convoluzionale, il codificatore trasmette solo i bit di parità.
- ▶ Ad ogni tempo di bit, il codificatore combina i bit all'interno di una finestra mobile di lunghezza L (*constraint length* del codice) per generare gli n bit in uscita.
- ▶ Il processo di codifica può essere interpretato come una convoluzione in $\text{GF}(2)$ ed il codificatore di un codice convoluzionale (n, k, L) si può rappresentare come n filtri lineari in $\text{GF}(2)$ in parallelo.

- Trasmette messaggio più parità non solo parità.

Introduzione ai codici convoluzionali

Un *codificatore convoluzionale* $\mathcal{C}(n, k, L)$ è costituito da uno shift register e da n sommatori.

- ▶ Ad ogni periodo di clock (definito dall'intero i) nel codificatore entrano k (tipicamente $k = 1$) cifre binarie rappresentate dal vettore $\mathbf{u}^{(i)} = [u_1^{(i)}, u_2^{(i)}, \dots, u_k^{(i)}]$ ed escono n bit, raccolti nel vettore di uscita $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}]$.
- ▶ All'interno del codificatore, $\mathbf{u}^{(i)}$ è combinato con i precedenti $L - 1$ vettori di ingresso $\mathbf{u}^{(i-1)}, \mathbf{u}^{(i-2)}, \dots, \mathbf{u}^{(i-L+1)}$ per formare l'uscita $\mathbf{x}^{(i)}$.



Introduzione ai codici convoluzionali

La principale differenza rispetto ai codici a blocco è costituita dal fatto che l'uscita attuale non dipende solo dalla parola attuale ma anche dalla storia passata contenuta nelle $L - 1$ parole precedenti.

- ▶ Il numero L di parole che contribuiscono all'uscita attuale è la *constraint length* del codice.
- ▶ Il numero complessivo di celle di memoria del codificatore è $kL - 1$: $(L - 1) \times k$ per memorizzare la memoria del sistema più $k - 1$ per memorizzare i bit (tutti eccetto l'ultimo) della parola attuale.

L'ultimo bit non è messo in memoria e per questo $k - 1$.

$$\begin{aligned}(L - 1) \cdot k + (k - 1) &= \\ (L-1+1) \cdot k - 1 &= \\ kL - 1\end{aligned}$$

8.2 Generatori

I generatori di un codice convoluzionale

Dato un codice convoluzionale $\mathcal{C}(n, k, L)$, i generatori definiscono completamente il codice, così come la matrice generatrice o il polinomio generatore definiscono i codici a blocco e i codici ciclici.

- ▶ A ciascuna delle n uscite corrisponde un generatore e a ciascun generatore corrisponde un sommatore in GF(2).
- ▶ Ciascun generatore ha kL elementi, uno per ciascun ingresso che contribuisce all'uscita del codificatore.
- ▶ Quando un elemento è 1 significa che il corrispondente ingresso è connesso al sommatore, altrimenti se è 0 non c'è connessione.

I generatori di un codice convoluzionale

- ▶ I vettori generatori rappresentano la risposta impulsiva degli n filtri lineari in GF(2).
- ▶ Assumendo che sia $k = 1$, il generatore per il bit m -esimo è $\mathbf{g}_m = [g_m^{(0)}, g_m^{(1)}, \dots, g_m^{(L-1)}]$ e l' m -esimo bit di uscita $x_m^{(i)}$ si ottiene così

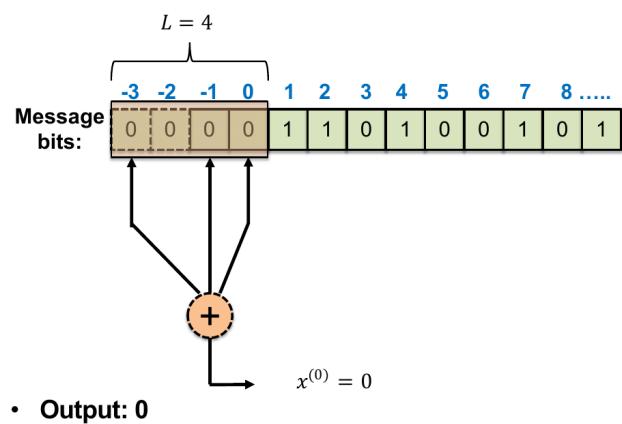
$$x_m^{(i)} = \sum_{\ell=0}^{L-1} g_m^{(\ell)} u^{(i-\ell)}.$$

- ▶ L'uscita m -esima è la convoluzione discreta in GF(2) tra i vettori $[\mathbf{u}^{(i)}, \mathbf{u}^{(i-1)}, \dots, \mathbf{u}^{(i-L+1)}]$ e \mathbf{g}_m , da cui il nome di *codice convoluzionale*.

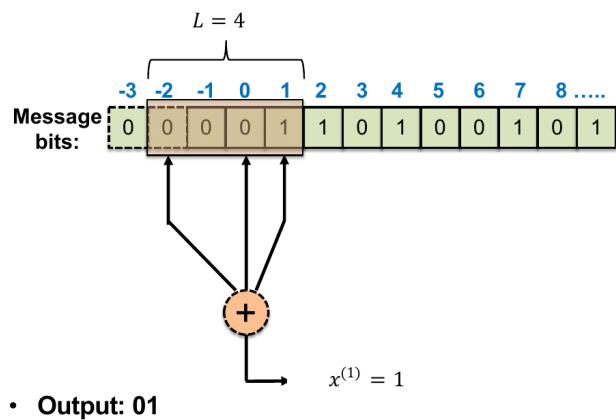
- $i \rightarrow$ indice temporale

8.2.1 Esempio di generatore

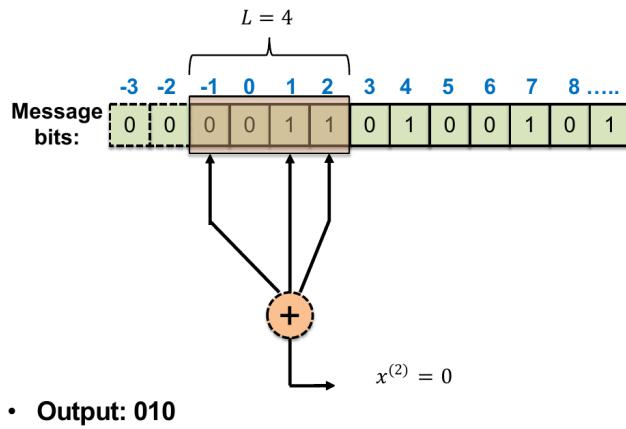
Esempio di generatore $\mathbf{g} = [1101]$



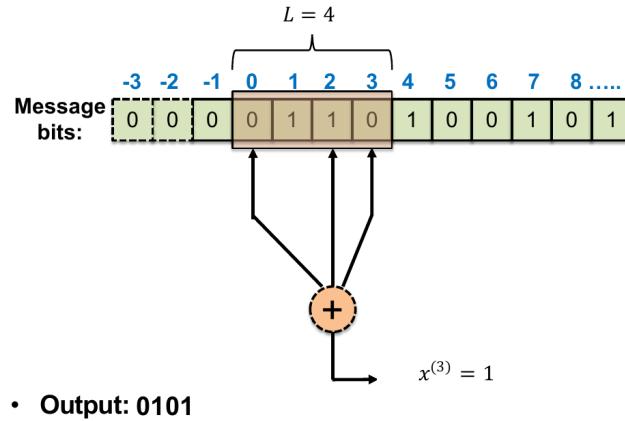
Esempio di generatore $\mathbf{g} = [1101]$



Esempio di generatore $\mathbf{g} = [1101]$



Esempio di generatore $\mathbf{g} = [1101]$



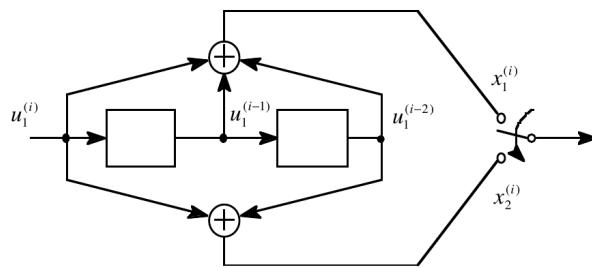
Esempio di codice convoluzionale: i generatori

Consideriamo il codice con $k = 1$ e $R = 1/2$, con generatori $\mathbf{g}_1 = [1, 1, 1] = 7_8$ e $\mathbf{g}_2 = [1, 0, 1] = 5_8$. La constraint length è $L = 3$.

- Le due uscite sono

$$\begin{cases} x_1^{(i)} = u_1^{(i)} + u_1^{(i-1)} + u_1^{(i-2)} \\ x_2^{(i)} = u_1^{(i)} + u_1^{(i-2)} \end{cases}$$

- Il diagramma a blocchi del codificatore è



- $7_8 \rightarrow 7$ in base 8

- L = dimensione dei generatori
 - Per ogni nuovo u in ingresso avendo due generatori g_1 e g_2 otteniamo due bit x in uscita e un nuovo stato.

Gli \emptyset in testa non si mettono quindi per esempio:

[0 0 0 1 0 1]

Diventa:

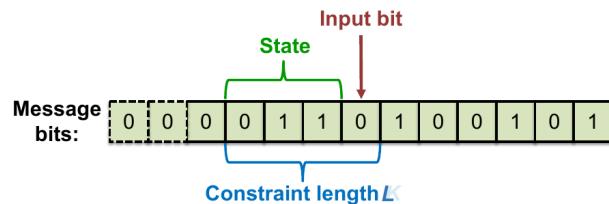
$$\begin{bmatrix} 1 & 0 & 1 \end{bmatrix}$$

8.2.2 Macchina a stati

Rappresentazione di un codice convoluzionale come macchina a stati

- ▶ Ad ogni istante di segnalazione l'uscita del codificatore convoluzionale dipende dall'ingresso attuale e dalla memoria.
 - ▶ Le ultime $k(L - 1)$ celle dello shift register memorizzano la storia passata del codificatore che è riassunta nel vettore di stato

$$\sigma^{(i)} = \left(\mathbf{u}^{(i-1)}, \mathbf{u}^{(i-2)}, \dots, \mathbf{u}^{(i-L+1)} \right)$$



Rappresentazione di un codice convoluzionale come macchina a stati

- Il codificatore può pensarsi come una *macchina a stati finiti* la cui evoluzione nel tempo è descritta da equazioni di stato

$$\begin{cases} \mathbf{x}^{(i)} = \delta(\mathbf{u}^{(i)}, \boldsymbol{\sigma}^{(i)}) \\ \boldsymbol{\sigma}^{(i+1)} = \lambda(\mathbf{u}^{(i)}, \boldsymbol{\sigma}^{(i)}) \end{cases}$$

La prima è detta *equazione di uscita*, la seconda *equazione di transizione di stato*. Il numero di stati complessivo è $2^{k(L-1)}$.

- Il funzionamento del codificatore si può esprimere con un *diagramma di stato* nel quale sono indicate le transizioni da uno stato all'altro sotto l'effetto dei diversi ingressi.

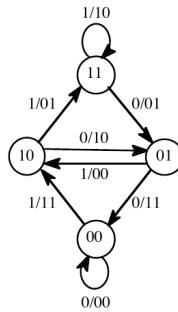
- $k(L-1) \rightarrow$ numero di bit nelle celle di memoria

Esempio di codice convoluzionale: il diagramma di stato

Consideriamo il codice con $k = 1$ e $R = 1/2$, con generatori

$\mathbf{g}_1 = [1, 1, 1] = 7_8$ e $\mathbf{g}_2 = [1, 0, 1] = 5_8$.

- Il numero di stati è $2^{k(L-1)} = 4$ ed il diagramma di stato è



- Nel cerchio abbiamo lo stato attuale delle celle
- Sopra gli archi abbiamo:

ingresso / nuovo stato $x_1 \ x_2$

Introduzione ai codici convoluzionali

- ▶ Il diagramma di stato non ha un indicatore temporale.
Introducendo un indicatore temporale il diagramma di stato si trasforma nel *diagramma a traliccio*, su cui è possibile seguire l'evoluzione degli stati e delle uscite del codificatore in seguito a una determinata sequenza di vettori di ingresso.
- ▶ Partendo da un qualsiasi stato, si può raggiungere qualsiasi altro stato del traliccio in un numero massimo di $L - 1$ passi.

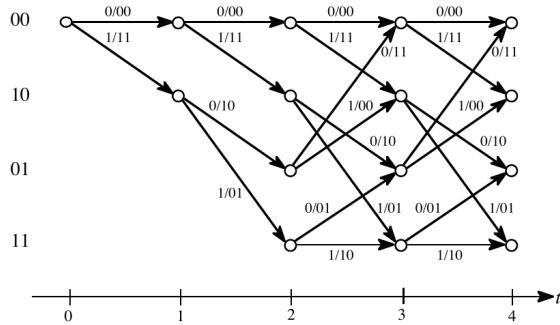


- Dopo $L-1$ passi tutti i bit sono stati cambiati in memoria e quindi si può ottenere qualsiasi altro stato.

8.3 Diagramma a traliccio

Esempio di codice convoluzionale: il diagramma a traliccio

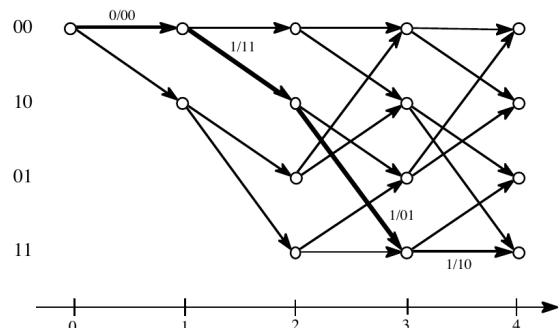
- Il diagramma a traliccio per il codice con $k = 1$ e $R = 1/2$, con generatori $\mathbf{g}_1 = [1, 1, 1] = 7_8$ e $\mathbf{g}_2 = [1, 0, 1] = 5_8$ è



Esempio di codice convoluzionale: il diagramma a traliccio

Il diagramma a traliccio permette di osservare l'evoluzione del codificatore in risposta ad una determinata sequenza di ingresso.

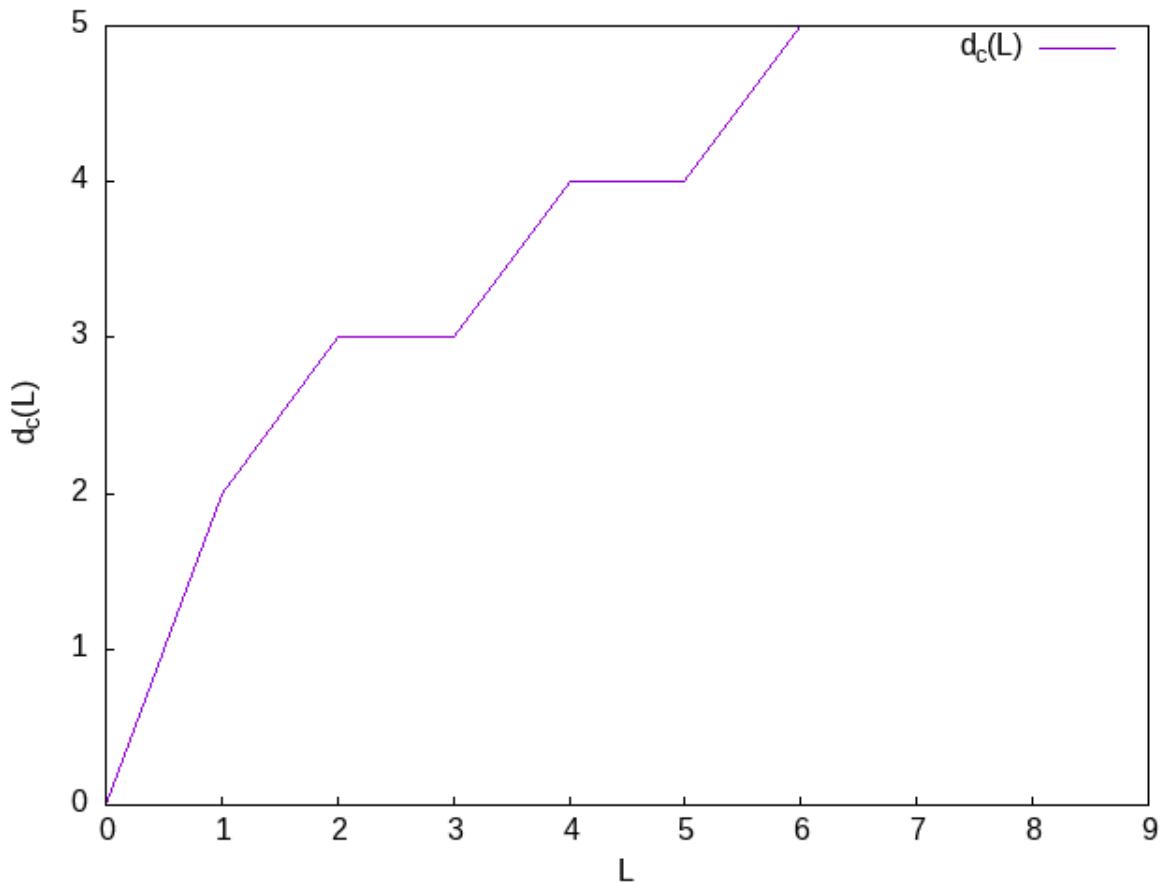
Per esempio per il codice con $k = 1$ e generatori $\mathbf{g}_1 = 7_8$ e $\mathbf{g}_2 = 5_8$ se $\sigma^{(0)} = 00$ e in ingresso si ha la sequenza $u^{(0)} = 0$, $u^{(1)} = 1$, $u^{(2)} = 1$ e $u^{(3)} = 1$ il percorso sul traliccio sarà



La distanza colonna in questo traliccio calcolata per vari l è:

$d_c(l)$	1 = 1	1 = 2	1 = 3	1 = 4	1 = 5	1 = 6	1 = 7	1 = 8	1 = ...
	2	3	3	4	4	5	5	5	5

La distanza colonna si rappresenta anche come un grafico:



8.3.1 Distanza colonna

Distanza colonna per un codice convoluzionale

- ▶ Sia dato un codice convoluzionale $\mathcal{C}(n, k, L)$. Consideriamo $\mathcal{X}_{\mathcal{C}}(\ell, \sigma)$, l'insieme di tutte le possibili sequenze che originano dallo stato σ e hanno lunghezza pari a ℓn , ottenute cioè dopo ℓ passi sul traliccio.
- ▶ La *distanza colonna* del codice \mathcal{C} al passo ℓ è la minima distanza di Hamming fra due sequenze \mathbf{x} e \mathbf{x}' in $\mathcal{X}_{\mathcal{C}}(\ell, \sigma)$ che siano differenti nei primi n bit, in corrispondenza quindi della prima parola di uscita del codificatore

$$d_c(\ell) = \min_{\substack{\mathbf{x}, \mathbf{x}' \in \mathcal{X}_{\mathcal{C}}(\ell, \sigma) \\ \mathbf{x}^{(1)} \neq \mathbf{x}'^{(1)}}} d_H(\mathbf{x}, \mathbf{x}')$$

- ▶ Come nel caso del calcolo della d_{min} di un codice a blocco, non fa nessuna differenza la particolare sequenza di riferimento, e quindi nel calcolo della $d_c(\ell)$ si sceglie lo stato σ in modo che una delle due sequenze sia il vettore di tutti zeri.

8.4 Distanza libera

Distanza libera per un codice convoluzionale

- ▶ La distanza libera di un codice convoluzionale \mathcal{C} , d_{free} è il limite della distanza colonna per ℓ che tende all'infinito, i.e,

$$d_{\text{free}} = \lim_{\ell \rightarrow \infty} d_c(\ell)$$

- ▶ In pratica la distanza libera coincide con la distanza colonna quando i due percorsi sul traliccio a massima distanza confluiscono.
- ▶ La distanza libera, come la d_{\min} nei codici a blocco, è una misura della bontà del codice: tanto maggiore è la distanza libera tanto più è difficile confondere due sequenze che originano dallo stesso stato.

8.5 Decodifica a massima verosimiglianza

Strategia di decodifica a massima verosimiglianza

- ▶ Si consideri una trasmissione su N intervalli di segnalazione. La trasmissione è codificata con il codice convoluzionale $\mathcal{C}(n, k, L)$, che ad ogni segnalazione associa k bit di informazione a n bit codificati.
 - ▶ Sia \mathbf{y} la sequenza ricevuta di lunghezza nN bit, la strategia di decodifica a massima verosimiglianza consiste nel trovare la sequenza $\hat{\mathbf{x}}$ che, fra tutte le 2^{kN} possibili sequenze di tentativo $\tilde{\mathbf{x}}$, massimizza la probabilità condizionata $P(\mathbf{y}|\tilde{\mathbf{x}})$, ie

$$\hat{\mathbf{x}} = \arg \max_{\tilde{\mathbf{x}}} P(\mathbf{y}|\tilde{\mathbf{x}})$$

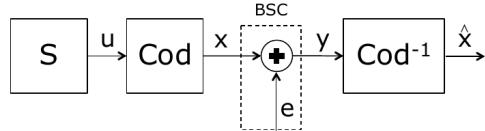
- ▶ Poichè gli eventi di errore sono indipendenti da bit a bit è conveniente riscrivere la probabilità condizionata come il prodotto delle probabilità condizionate ottenute per ciascuna parola di codice trasmessa

$$P(\mathbf{y}|\tilde{\mathbf{x}}) = \prod_{\ell=1}^N P\left(\mathbf{y}^{(\ell)}|\tilde{\mathbf{x}}\right) = \prod_{\ell=1}^N P\left(\mathbf{y}^{(\ell)}|\tilde{\mathbf{x}}^{(\ell)}\right) \quad (1)$$

- ... k bit di informazione ... = ... k bit di parità ...
 - Le possibili combinazioni derivate dai bit di parità sono 2^{kN} .

Strategia di decodifica a massima verosimiglianza

- ▶ Si assume che la probabilità di ricevere un bit errato sia p .



- ▶ La probabilità condizionata $P(\mathbf{y}^{(\ell)}|\tilde{\mathbf{x}}^{(\ell)})$ si ottiene a partire dal calcolo della distanza di Hamming $\lambda_{\tilde{\mathbf{x}}}^{(\ell)} = d_H(\mathbf{y}^{(\ell)}, \tilde{\mathbf{x}}^{(\ell)})$, infatti per ogni bit diverso tra le due parole si assume che ci sia stato un errore e quindi

$$P\left(\mathbf{y}^{(\ell)}|\tilde{\mathbf{x}}^{(\ell)}\right) = p^{\lambda_{\tilde{\mathbf{x}}}^{(\ell)}}(1-p)^{n-\lambda_{\tilde{\mathbf{x}}}^{(\ell)}} = (1-p)^n \left(\frac{p}{1-p}\right)^{\lambda_{\tilde{\mathbf{x}}}^{(\ell)}} \quad (2)$$

Strategia di decodifica a massima verosimiglianza

- ▶ Poiché la funzione logaritmo è crescente, la sequenza di tentativo che massimizza $P(\mathbf{y}|\tilde{\mathbf{x}})$ massimizza anche $\log(P(\mathbf{y}|\tilde{\mathbf{x}}))$.
 - ▶ Sostituendo in (1) il valore trovato in (2) e prendendo il logaritmo del risultato si ottiene

$$\log(P(\mathbf{y}|\tilde{\mathbf{x}})) = nN \log(1-p) + \log\left(\frac{p}{1-p}\right) \sum_{\ell=1}^N \lambda_{\tilde{\mathbf{x}}}^{(\ell)}$$

- ▶ Trascurando i termini ininfluenti e considerando che è lecito assumere $\log\left(\frac{P}{1-P}\right) < 0$, si ottiene la regola di decisione

$$\hat{\mathbf{x}} = \arg \max_{\tilde{\mathbf{x}}} P(\mathbf{y}|\tilde{\mathbf{x}}) = \arg \min_{\tilde{\mathbf{x}}} \sum_{\ell=1}^N \lambda_{\tilde{\mathbf{x}}}^{(\ell)}$$

- La sommatoria dei lambda è la distanza tra la sequenza ipotizzata e quella ricevuta.

Strategia di decodifica a massima verosimiglianza

- ▶ La strategia a massima verosimiglianza consiste nello scegliere la sequenza $\hat{\mathbf{x}}$ che minimizza la distanza di Hamming dalla sequenza ricevuta, i.e,

$$d_H(\mathbf{y}, \hat{\mathbf{x}}) = \sum_{\ell=1}^N \lambda_{\hat{\mathbf{x}}}^{(\ell)}$$

- ▶ Se si osserva che la trasmissione dura N intervalli di segnalazione, in cui ogni volta vengono inviati al codificatore k bit, il numero di sequenza da considerare è 2^{kN} .
- ▶ ATTENZIONE: La complessità della decodifica cresce *esponenzialmente* con k e N .

8.6 Algoritmo di Viterbi

Algoritmo di Viterbi

- ▶ Consideriamo un codice $\mathcal{C}(n, k, L)$ e assumiamo che si effettui una trasmissione di N parole di codice ciascuna composta da n bit e che lo stato iniziale $\sigma^{(0)}$ e quello finale $\sigma^{(N)}$ del codificatore siano conosciuti anche al ricevitore.
- ▶ Data la sequenza ricevuta \mathbf{y} di nN bit, l'obiettivo del decodificatore è individuare sul traliccio il percorso $\hat{\mathbf{x}}$ più breve (a minima distanza di Hamming!) che partendo da $\sigma^{(0)}$ arrivi a $\sigma^{(N)}$.
- ▶ La sequenza $\hat{\mathbf{x}}$ deve minimizzare la metrica

$$d_H(\mathbf{y}, \hat{\mathbf{x}}) = \Lambda(\mathbf{y}, \hat{\mathbf{x}}) = \sum_{\ell=1}^N \lambda_{\hat{\mathbf{x}}}^{(\ell)}$$

- L'ipotesi sui sigma vuol dire che si mette in ingresso e un uscita un flag conosciuto da trasmettitore e ricevitore, per esempio: 1111

Algoritmo di Viterbi

L'algoritmo a bassa complessità per trovare il percorso più breve sul traliccio è stato brevettato nel 1967 da A.J. Viterbi.

- ▶ L'intuizione fondamentale è che un gran numero dei 2^{kN} possibili percorsi sul traliccio non è rilevante al fine del calcolo della distanza minima e quindi può essere scartato.



Andrew James Viterbi
(born Andrea Giacomo Viterbi
in Bergamo, Italy)
is Professor of Electrical
Engineering at the University of
Southern California's Viterbi
School of Engineering, which
was named after him in
recognition of his \$52 million
gift.

Algoritmo di Viterbi

Si definisce:

- ▶ *metrica di ramo* $\sigma_j \rightarrow \sigma_k$ al passo ℓ , la distanza di Hamming $\lambda^{(\ell)}(\sigma_j, \sigma_k) = d_H(\mathbf{y}^{(\ell)}, \mathbf{x}_{\sigma_j \rightarrow \sigma_k})$. La metrica è calcolata come la distanza tra la sequenza ricevuta al passo ℓ e l'uscita corrispondente alla transizione sul traliccio dallo stato σ_j allo stato σ_k ;
- ▶ *metrica cumulata* al passo f allo stato σ_k , la grandezza $\Lambda_x^{(f)}(\sigma_k)$ ottenuta sommando tutte le f metriche di ramo calcolate su gli f rami sul traliccio di un percorso \mathbf{x} che si ferma allo stato σ_k al passo f .

- $\sigma_j \rightarrow \sigma_k$ indica una singola freccia nel traliccio.

- La metrica cumulata calcola in un certo senso il numero di errori di un ramo che arriva in un pallino del traliccio.

Algoritmo di Viterbi

- ▶ L'algoritmo di Viterbi si basa sulla seguente intuizione:
 - ▶ Supponiamo che due diversi percorsi x_1 e x_2 confluiscono al passo f nello stesso nodo σ_k sul traliccio e siano $\Lambda_{x_1}^{(f)}(\sigma_k) < \Lambda_{x_2}^{(f)}(\sigma_k)$ le metriche cumulate dei due percorsi calcolate al passo f .
 - ▶ Supponiamo che al passo successivo $f+1$, i percorsi x_1 e x_2 seguano lo stesso ramo sul traliccio, ad esempio $\sigma_j \rightarrow \sigma_q$. In questo caso la metrica di ramo è la stessa per i due percorsi $\lambda^{(f+1)}(\sigma_k, \sigma_q)$.
 - ▶ Per le nuove metriche sarà ancora $\Lambda_{x_1}^{(f+1)}(\sigma_q) < \Lambda_{x_2}^{(f+1)}(\sigma_q)$, infatti si ha

$$\begin{aligned}\Lambda_{x_1}^{(f+1)}(\sigma_q) &= \Lambda_{x_1}^{(f)}(\sigma_k) + \lambda^{(f+1)}(\sigma_k, \sigma_q) \\ &< \Lambda_{x_2}^{(f)}(\sigma_k) + \lambda^{(f+1)}(\sigma_k, \sigma_q) = \Lambda_{x_2}^{(f+1)}(\sigma_q)\end{aligned}$$

Algoritmo di Viterbi

Generalizzando questa intuizione si può dire che :

1. In un traliccio arrivato alla sua piena espansione ad ogni nuovo istante di segnalazione arrivano 2^k percorsi a ciascun nodo.
2. I valori delle metriche di ramo in uscita da un nodo sono uguali per tutti i percorsi che arrivano a quel nodo.
3. A parità di percorso futuro, il percorso in ingresso al nodo con la metrica cumulata minore continuerà ad avere metrica cumulata più bassa di tutti gli altri.
4. Ai fini della minimizzazione della metrica cumulata complessiva ad ogni nodo si possono scartare i $2^k - 1$ percorsi con la metrica cumulata più alta e conservare solo quello con la metrica cumulata minima.
5. L'unico percorso rimasto viene definito *sopravvissuto*.

8.6.1 Esempio di applicazione di Viterbi

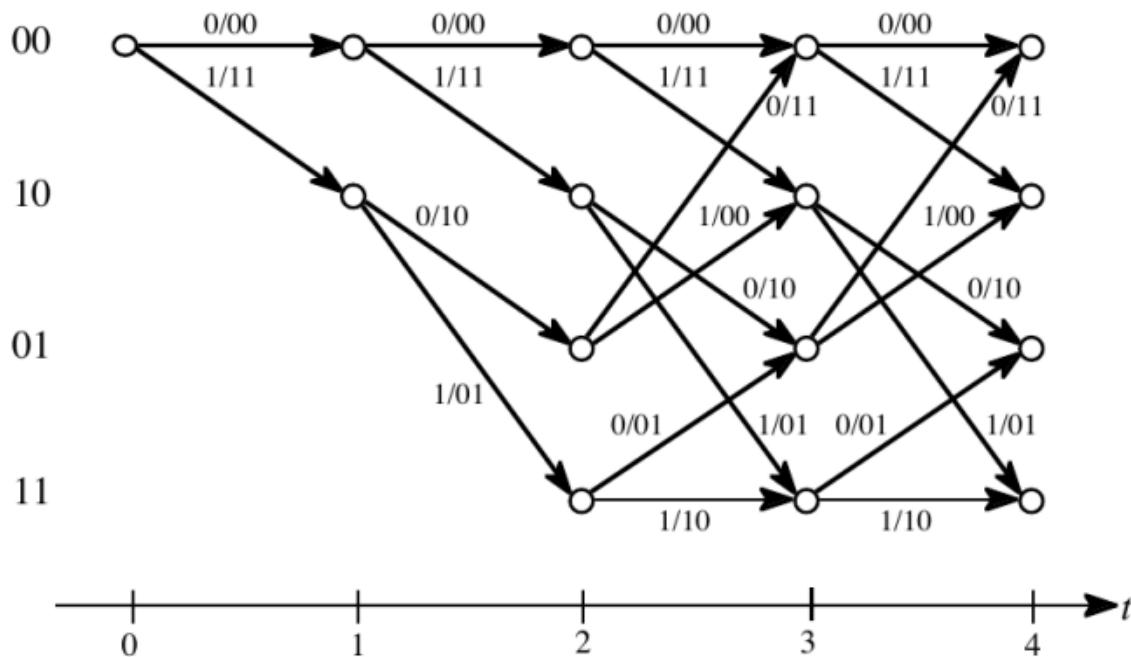
Esempio di applicazione dell'algoritmo di Viterbi

- ▶ Consideriamo il codice convoluzionale con rate $R = 1/2$, constraint length $L = 3$ e generatori $\mathbf{g}_1 = 7_8$ e $\mathbf{g}_2 = 5_8$.
- ▶ Assumiamo che la sequenza di $N = 6$ bit informativi sia $\mathbf{u} = [0, 1, 0, 0, 0, 0]$ a cui corrisponde la sequenza codificata $\mathbf{x} = [00, 11, 10, 11, 00, 00]$ e supponiamo che la sequenza ricevuta sia $\mathbf{y} = [00, 1\textcolor{red}{0}, 10, 11, 00, 00]$, in cui il 2o bit della 2a parola di codice è errato.
- ▶ In un sistema reale, $N \gg 6$, $N = 6$ è solo a scopo illustrativo.

Esempio di applicazione dell'algoritmo di Viterbi

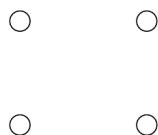
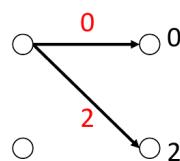
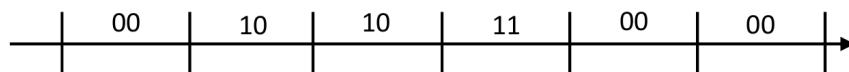
- ▶ Le metriche di ramo sono indicate in rosso, quelle cumulate in nero.
- ▶ La trasmissione inizia nello stato $\sigma^{(0)} = [0, 0]$ e dopo $N = 6$ segnalazioni torna nello stato $\sigma^{(N)} = [0, 0]$.

Mentre segui l'esempio tieniti di fianco il traliccio:

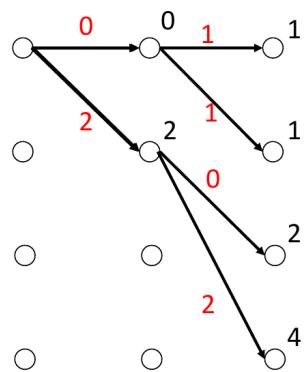
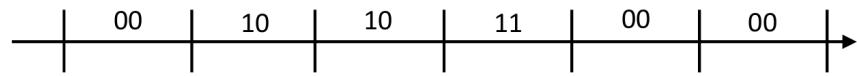


- Il numero rosso è la distanza di Hamming tra i bit decodificati(ovvero quelli a destra dello slash 0/questi) e i bit di y al relativo passo che li vedi nella "tabella" in alto nello svolgimento dell'esercizio. Le cifre nere sono la somma delle rosse seguendo un particolare percorso.

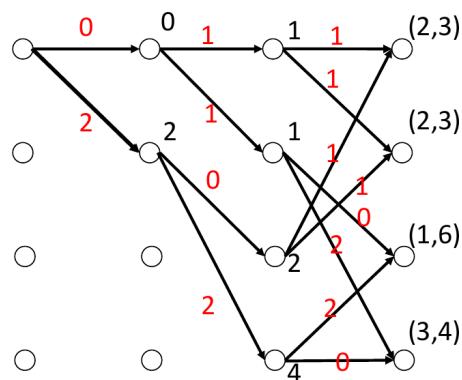
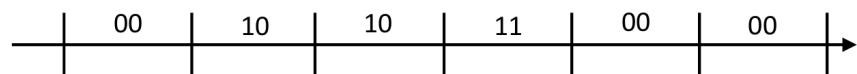
Esempio di applicazione dell'algoritmo di Viterbi



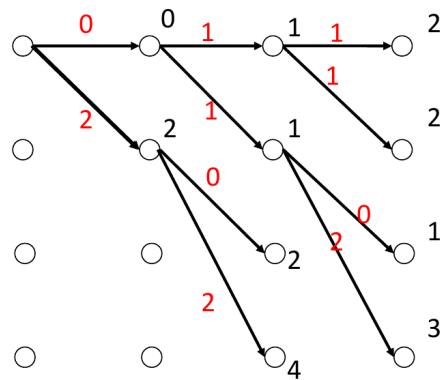
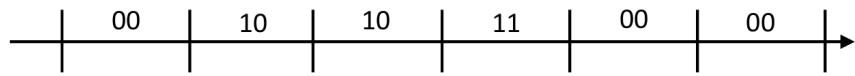
Esempio di applicazione dell'algoritmo di Viterbi



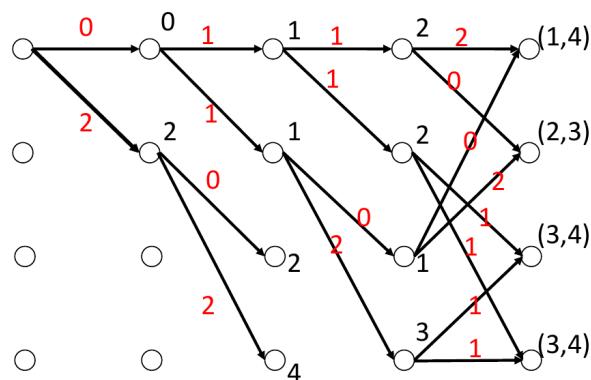
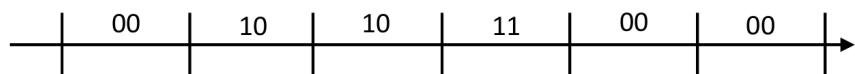
Esempio di applicazione dell'algoritmo di Viterbi



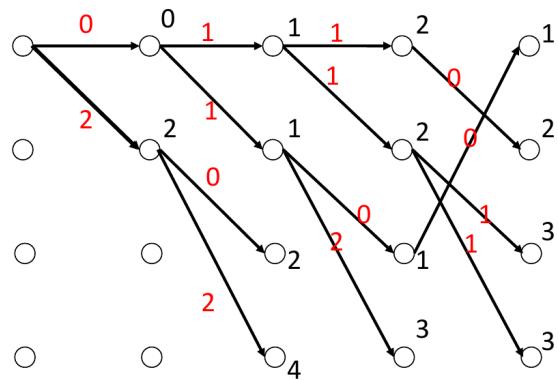
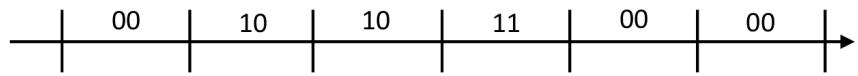
Esempio di applicazione dell'algoritmo di Viterbi



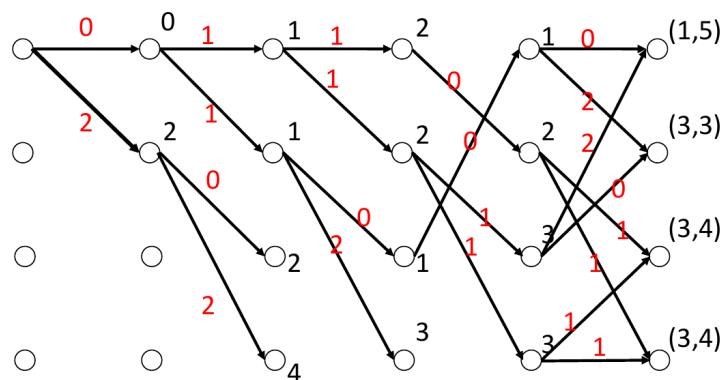
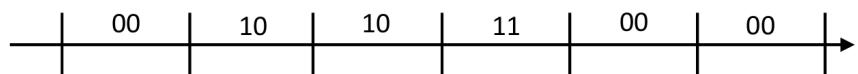
Esempio di applicazione dell'algoritmo di Viterbi



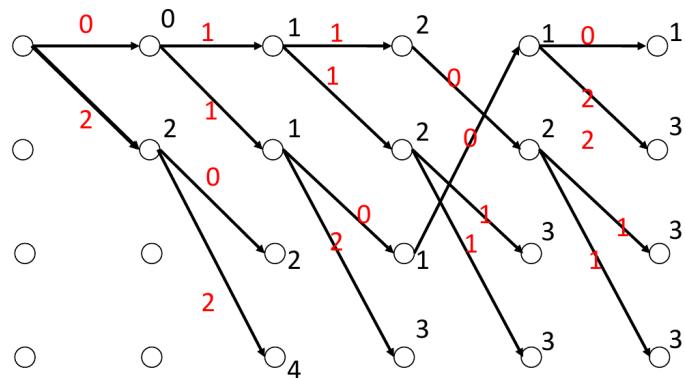
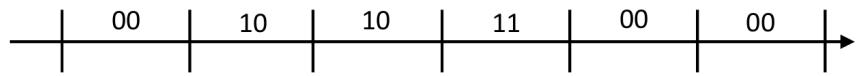
Esempio di applicazione dell'algoritmo di Viterbi



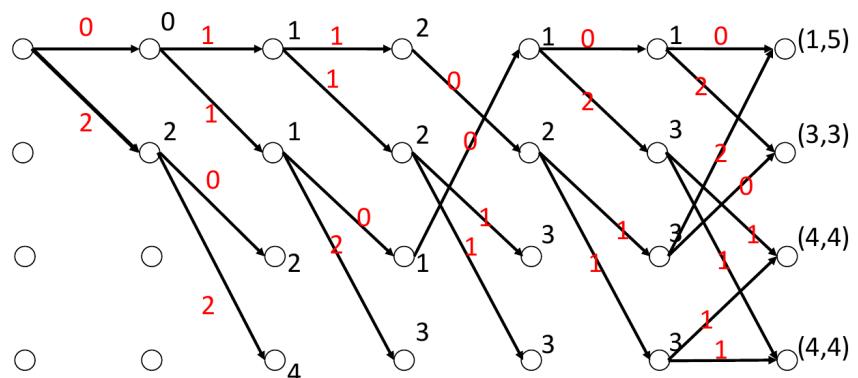
Esempio di applicazione dell'algoritmo di Viterbi



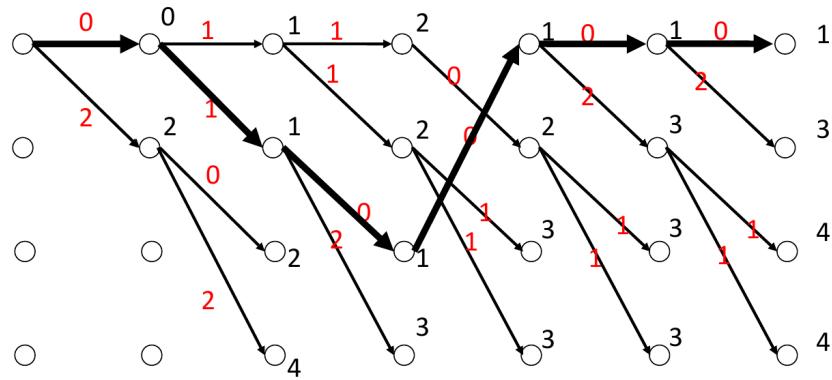
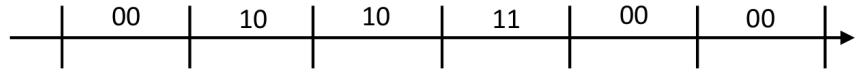
Esempio di applicazione dell'algoritmo di Viterbi



Esempio di applicazione dell'algoritmo di Viterbi



Esempio di applicazione dell'algoritmo di Viterbi



Esempio di applicazione dell'algoritmo di Viterbi

- ▶ La trasmissione inizia nello stato 00 e dopo $N = 6$ segnalazioni torna nello stato 00.
- ▶ Una volta identificato il percorso a distanza minima sul traliccio, identificare la sequenza $\hat{\mathbf{u}}$ informativa è semplice: è sufficiente associare ad ogni transizione sul traliccio la sequenza di bit di ingresso corrispondente.
- ▶ Nel nostro caso si ha $\hat{\mathbf{u}} = [0, 1, 0, 0, 0, 0] = \mathbf{u}$. Il decodificatore ha corretto l'errore introdotto dal canale!

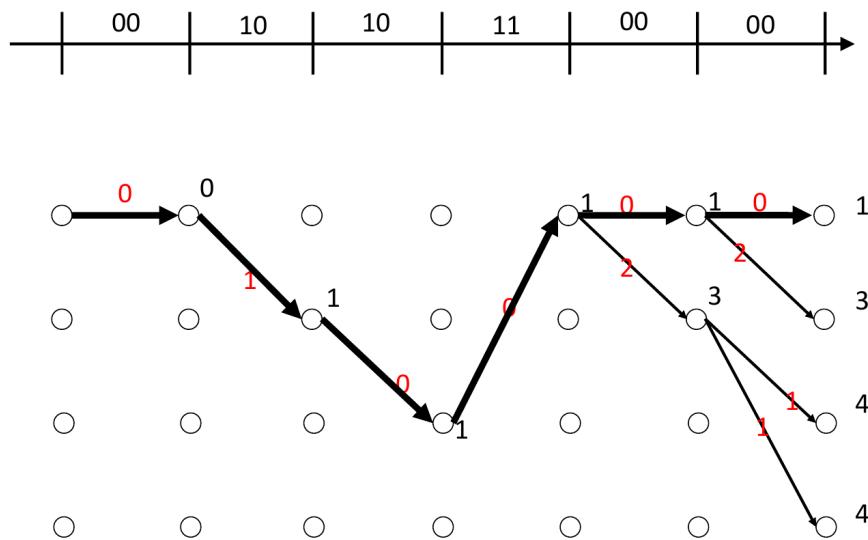
8.6.2 Considerazioni algoritmo di Viterbi

Considerazioni finali sull'algoritmo di Viterbi

- ▶ A partire dal momento in cui il traliccio ha raggiunto il suo pieno sviluppo, ad ogni passo l'algoritmo deve
 1. Calcolare le metriche di ramo per tutte le transizioni possibili;
 2. Calcolare per ogni nodo le metriche cumulate dei percorsi in ingresso;
 3. Per ogni nodo trovare il percorso sopravvissuto associato alla metrica cumulata minima.
 - ▶ La complessità dell'algoritmo di Viterbi cresce *linearmente* con la lunghezza della sequenza da decodificare e non esponenzialmente, come avviene con la ricerca esaustiva.



Considerazioni finali sull'algoritmo di Viterbi



Considerazioni finali sull'algoritmo di Viterbi

Osservando il traliccio al passo ℓ , si trova che andando a ritroso di D passi, i sopravvissuti tendono a fondersi in un unico percorso.

Poiché si è visto empiricamente che D è una quantità fissa dell'ordine di kL , possiamo trarre le seguenti conclusioni:

1. Il decodificatore di Viterbi non deve necessariamente attendere di essere arrivato alla fine della segnalazione per produrre decisioni, ma è sufficiente introdurre un ritardo di decodifica pari a D e poi prendere decisioni di decodifica ad ogni nuovo istante di segnalazione.
2. La parte comune dei sopravvissuti (quella su cui sono già state prese le decisioni) non ha bisogno di essere memorizzata. È sufficiente memorizzare i sopravvissuti nell'ultimo tratto da $\ell - D$ a ℓ . Questo riduce enormemente l'esigenza di memoria dell'algoritmo.



- TODOS

- Disegno delle sequenze
- Video per spiegazione distanza libera
- Video per (1)
- Algoritmo di Viterbi 2

9 Calcolo della probabilità di errore sulle parole di codice

Calcolo della probabilità di errore sulle parole di codice

Un codice a blocco $\mathcal{C}(k, n)$ con $d_{min} = 2t + 1$ è in grado di correggere fino a t errori.

- ▶ Una parola ricevuta $\mathbf{y} = \mathbf{x} + \mathbf{e}$ è errata quando il canale introduce un numero di errori maggiore di t .
 - ▶ La probabilità di errore $P_w(\mathbf{e}) = \Pr\{\mathbf{w}(\mathbf{e}) > t\}$ si calcola

$$P_w(e) = \sum_{j=t+1}^n \binom{n}{j} p^j (1-p)^{n-j}$$

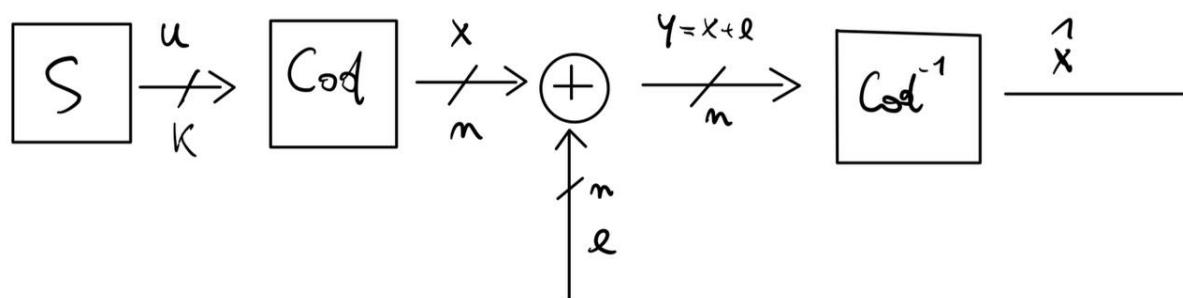
- ▶ $P_w(e)$ può essere lower-bounded dalla probabilità dell'evento più probabile: aver commesso $t + 1$ errori

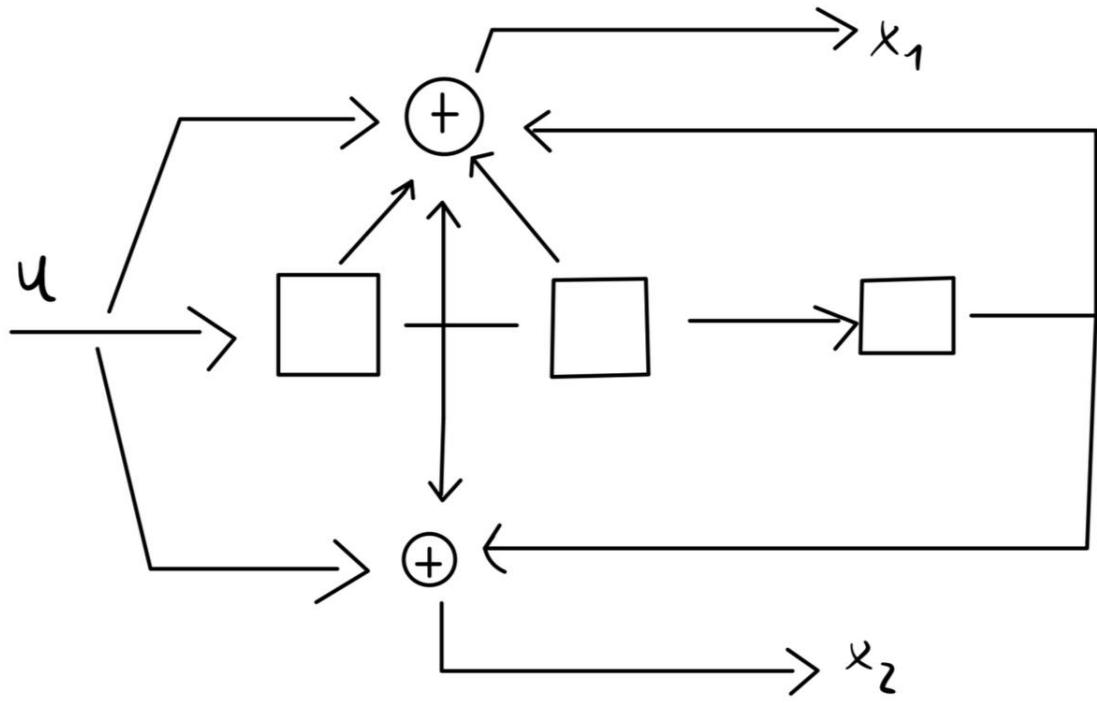
$$P_w(e) \approx \binom{n}{t+1} p^{t+1} (1-p)^{n-(t+1)}$$

- $P_w(e)$ è difficile da col calcolare e quindi la dovrò calcolare sul singolo bit.

Il nostro obiettivo è trovare una probabilità di errore sul bit in un sistema codificato. Per un sistema non codificato già la abbiamo:

$$2\text{-PAM} \Rightarrow Q\left(\sqrt{\frac{2E_0}{N_0}}\right)$$





Un errore al ricevitore ovvero la prima formula della slide si ha molto probabilmente quando $w(e) > t$. La probabilità di questo evento è:

$$P_r\{w(e) > t\} =$$

Questo perché noi sappiamo che abbiamo una probabilità di errore sul bit pari a:

- $P_e = p \rightarrow$ probabilità di bit errati
- $P_c = 1-p \rightarrow$ probabilità di bit corretti

Poi c'è anche da considerare un certo numero di errori $l > t$ tale che gli errori sono:

$$\binom{n}{l} p^l (1-p)^{n-l}$$

- $\binom{n}{l}$ sono le possibili combinazioni di l errori su n bit.

$$\binom{n}{l} = \frac{n!}{(n-l)!l!}$$

Ora la probabilità di errore la trovo nella slide. Se noi assumiamo che la probabilità di errore (p^j) sia abbastanza bassa posso approssimarla con $j = t+1$. Tenendo conto che p è un valore < 1 aumentando l'esponente abbiamo che la probabilità di errore p_e diminuisce. Possiamo quindi considerare il valore minimo dell'esponente l'evento più probabile andando a porre un lower bound ($t+1$) sulla probabilità di errore. Ricordiamo che a noi non interessa però la probabilità di errore sulla parola, ma la probabilità di errore sul bit. Ora tenendo presente questo ipotesi:

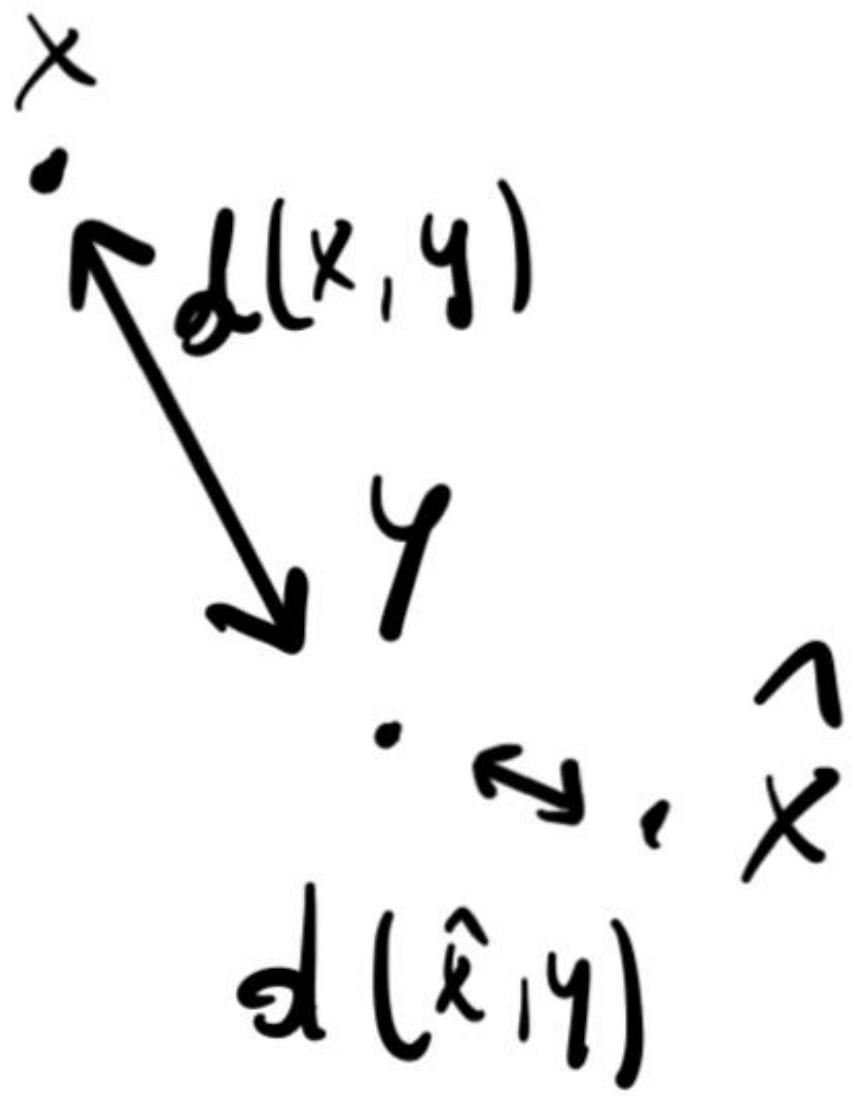
Bound per il calcolo della probabilità di errore sul bit

Mentre la $P_w(e)$ si riesce a calcolare con precisione, nel caso del calcolo della probabilità di errore su bit codificato si deve per forza ricorrere ad approssimazioni.

- ▶ Il numero di bit errati in \hat{x} dopo la decodifica dipende dal vettore di errore e e da come agisce la decodifica a sindrome, che, in presenza di un numero di errori maggiore di t , aggiunge altri errori a quelli introdotti dal canale.
- ▶ La decodifica a sindrome restituisce sempre una parola di codice, quindi ogni volta che al ricevitore c'è un errore nella decodifica i bit errati sono almeno d_{min} degli n trasmessi.
- ▶ In questo caso la $P_b(e)$ si approssima

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

Uno schema per capire perché la decodifica a sindrome introduce più errori in caso in cui nella parola ricevuta abbiamo un numero di errori maggiore di d_{min} è il seguente:



Ovvero qua si vede che la parola ricevuta è più vicina a un'altra parola di codice(y) rispetto a quella corretta(x). Vediamo ora un esempio:

$$\begin{aligned} x &= 1 \ 1 \ 1 \\ y &= 0 \ 1 \ 0 \quad w(e) = 2 \\ \hat{x} &= 0 \ 0 \ 0 \quad w(e) = 3 \end{aligned}$$

Abbiamo che la distanza di errore più probabile è d_{\min} e quindi il numero di bit sbagliati più probabile è d_{\min} . Ricordati del lower-bound. Notiamo che la probabilità di errore sul bit sarà minore di quella sulla parola.

9.1 Confronto delle prestazioni tra sistemi codificati e non

Confronto delle prestazioni tra sistemi codificati e non

La ridondanza introdotta dal codice comporta una maggiore 'spesa' energetica, infatti si utilizzano n bit codificati per trasmettere k bit di informazione.

- ▶ Il 'budget' energetico di k bit viene distribuito su n bit

$$kE_b = nE_{b,c} \implies E_{b,c} = \frac{k}{n}E_b$$

- A parità di energia per parola l'energia per bit codificato è minore.

$$E_{b,c} \rightarrow E_b \cdot \text{bit codificato}$$

Confronto delle prestazioni tra sistemi codificati e non

La probabilità di errore sul bit per una BPSK non codificata è

$$P_b^{(BPSK)}(e) = Q\left(\sqrt{\frac{2E_b}{N_0}}\right)$$

- ▶ Nel caso codificato bisogna considerare che la probabilità di errore p dipende dal valore di SNR dei bit codificati

$$\frac{E_{b,c}}{N_0} = \frac{k}{n} \frac{E_b}{N_0}$$

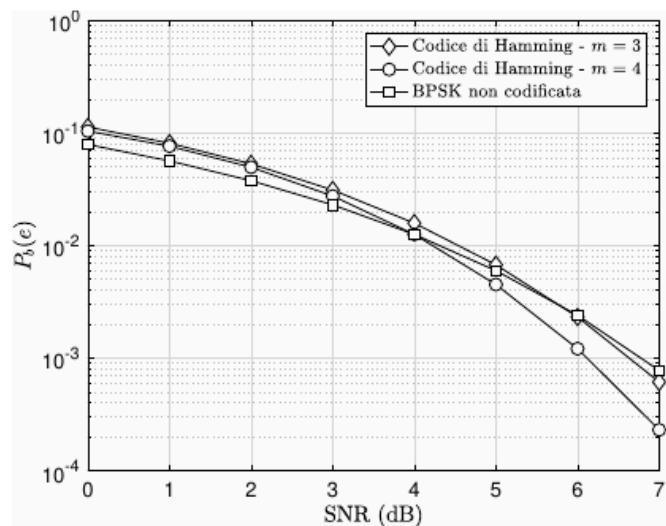
- ▶ La probabilità $P_b(e)$ del codice in (1), va calcolata utilizzando

$$p = Q\left(\sqrt{\frac{2E_{b,c}}{N_0}}\right) = Q\left(\sqrt{2 \frac{k}{n} \frac{E_b}{N_0}}\right). \quad (2)$$

- BPSK praticamente è un canale binario
- $N_0 \rightarrow$ rumore
- Nell'ultima formula sostituisco l'energia per bit trovata prima.

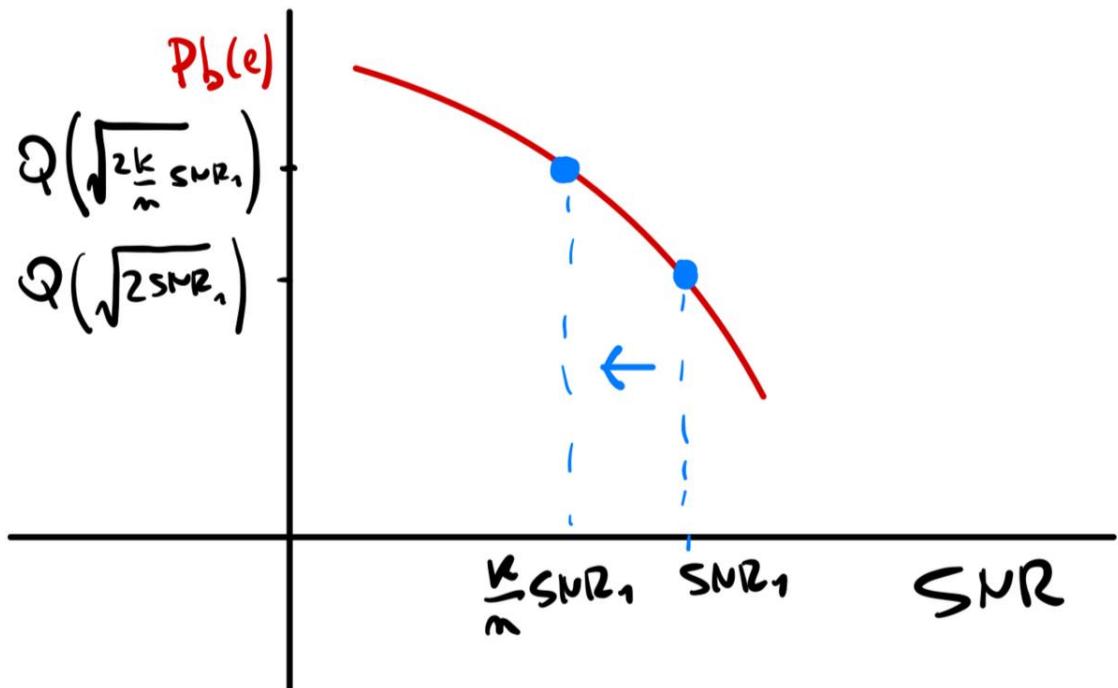
Confronto delle prestazioni tra sistemi codificati e non

- ▶ Confronto delle prestazioni su canale Gaussiano di un sistema BPSK senza codifica con le prestazioni di un sistema codificato con codice di Hamming con $m = 3$ e $m = 4$.



- Quadrato → probabilità di errore non codificata
 - Rombo → $m = 3$ $k = 4$, $n = 7$
 - Tondo → $m = 4$ $k = 11$, $n = 15$

Quando il rapporto SNR è molto basso ovvero tra 0 e 3 la probabilità di errore è grande e quindi è molto probabile avere un numero di errori maggiore di 1 contando che la d_{min} negli Hamming è 3. Quindi abbiamo che il codice non migliora le prestazioni, anzi potrebbe peggiorarle. Man mano che il rapporto SNR aumenta gli errori diventano meno probabili e il codice diventa migliore nelle sue capacità di correzione.



Introducendo un fattore $\frac{k}{n}$ dovuto alla codifica notiamo che per un certo SNR_1 la probabilità di errore aumenta e dal grafico si vede perché abbiamo uno spostamento del punto sulla curva Q a sinistra che quindi sale.

10 Codici Convoluzionali

Generatori per i codici convoluzionali

- ▶ La bontà di un codice convoluzionale dipende dalla sua d_{free} .
- ▶ La d_{free} dipende dai codici generatori, dal rate $R = k/n$ e dalla constraint length L .
- ▶ Tipicamente i codici convoluzionali hanno $k = 1$ per limitare la complessità di codificatore e decodificatore.
- ▶ Fissato R e L i generatori ottimi sono quelli che massimizzano la d_{free} e possono essere trovati tramite una ricerca esaustiva fra tutte le possibili $(2^L)^n = 2^{Ln}$ combinazioni.
- ▶ A causa della limitata complessità i codici convoluzionali a $R = 1/2$ sono quelli più studiati.



Siccome il codice diventa molto complesso all'aumentare di k solitamente si assume $k = 1$. Il rate quindi sarà sempre nella forma $\frac{1}{n}$. Attenzione che però non voglio introdurre troppa ridondanza. Tieni a mente che qua l'LSB è a destra mentre a blocco a sinistra osservando la seguente slide.

Generatori ottimi per $R = 1/2$

Generatori ottimi (in ottale!) per codici convoluzionali a rate $R = 1/2$ al variare della constraint length L e d_{free} corrispondente.

Constraint length	Generatori ottimi		Distanza libera
L	\mathbf{g}_1	\mathbf{g}_2	d_{free}
3	7	5	5
4	17	15	6
5	35	23	7
6	75	53	8
7	133	171	10
8	371	247	10
9	763	561	12
10	1537	1131	12

Il più usato è il 133 171 che è usato dalla Voyager.

10.1 Puncturing per codici convoluzionali

Puncturing per i codici convoluzionali

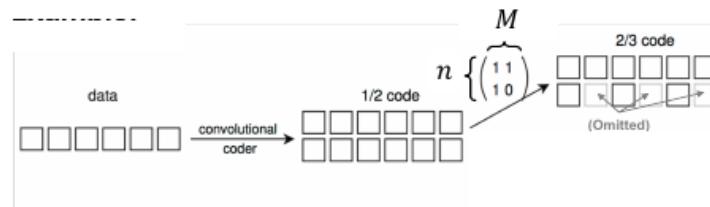
- ▶ In teoria, non c'è flessibilità nella scelta del rate dei codici convoluzionali che assume sempre valori del tipo $R = 1/n$.
- ▶ In realtà, la tecnica chiamata *puncturing* permette di costruire codici con rate maggiori partendo da un codice a rate $R = 1/n$.
- ▶ Il puncturing consiste nel cancellare alcuni bit all'uscita del codificatore. I bit vengono cancellati secondo un pattern preciso, espresso da una *puncturing table*, condiviso con il ricevitore, che quindi conosce esattamente la posizione dei bit cancellati.

Puncturing per i codici convoluzionali

- ▶ Il trasmettitore e il ricevitore si accordano sui bit codificati da omettere attraverso la puncturing table, che contiene n righe (una per bit in uscita) e M colonne. La matrice contiene un certo numero P di '1' e un numero $P - nM$ di '0'.
- ▶ Dopo il puncturing il rate del codice diventa

$$R' = \frac{1}{n} \frac{nM}{P} = \frac{M}{P}$$

- ▶ Esempio con $n = 2$, $M = 2$, $P = 3$ con rate $R' = 2/3$.



Chiamo P il numero di uni nella puncturing table e quindi il numero di bit che passano. Essenzialmente se dico che il rate $\frac{1}{2}$ è troppo basso e uso questa tecnica allora la d_{free} si riduce. Se $\frac{M}{P}$ è $\frac{1}{2}$ non cambia nulla nel codice e la d_{free} rimane uguale. Se già vado a $\frac{2}{3}$ ho una d_{free} più bassa. Tutte quelle che si vedono sono le puncturing tables ottime per i codici rappresentati.

Puncturing per i codici convoluzionali

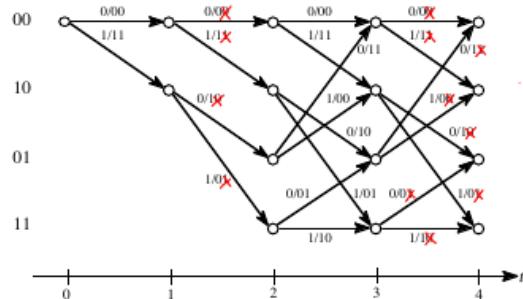
Tabella di puncturing per il codici convoluzionale a rate $R = 1/2, L = 7$ al variare della constraint length del rate $R = M/P$ in uscita e d_{free} corrispondente.
Il puncturing ottimo è stato trovato con una ricerca esaustiva su tutti i possibili pattern.

Rate M/P	Puncturing matrix	d_{free}
1/2	$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$	10
2/3	$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$	6
3/4	$\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$	5
5/6	$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}$	4
7/8	$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$	3

Esempio di traliccio dopo il puncturing

Traliccio del codice convoluzionale a rate $R = 2/3$ ottenuto dal codice convoluzionale ottimo con $R = 1/2$ e $L = 3$ applicando la matrice di puncturing [11; 10].

I bit in corrispondenza del puncturing (marcati con una croce rossa) non vengono trasmessi e al ricevitore non contribuiscono al calcolo delle metriche di ramo.



Bound per le prestazioni dei codici convoluzionali

Ad alti rapporti segnale-rumore si trova la seguente approssimazione

- ▶ BPSK codificata con decodifica hard

$$P_e^{(b)} \approx Q \left(\sqrt{2 \frac{E_b}{N_0} \frac{R d_{free}}{2}} \right)$$

- $R = \frac{K}{N}$
- Il $\frac{d_{free}}{2}$ mi sposta il punto a destra sulla Q