Università degli Studi di Bologna

DIPARTIMENTO DI MATEMATICA

Corso di Laurea Magistrale in Matematica

Tesi di laurea

Studio dei codici, trasmissione e correzione efficiente di un messaggio

Candidato:

Francesco Grigoli

Matricola 891459

Relatore:

Marco Lenci

Anno Accademico 2019–2020

Indice

Introduzione

La tecnologia è basata su alcuni principi matematici che le conferiscono il suo fondamento teorico. Questo vale per qualunque tipo di ingegneria: civile, aeronautica, informatica o elettronica. Tuttavia, nell'ora della verità, pochi conoscono questi principi. Il proposito di questa trattazione è, precisamente, mostrare i fondamenti sui quali si basa una delle tecnologie più usate nella vita quotidiana da centinaia di milioni di individui quasi senza pensarci: le telecomunicazioni.
Negli ultimi decenni la società ha sperimentato un notevole avanzamento tecnologico. Di fatto, al giorno d'oggi, secondo dati recenti, ogni secondo nel mondo nascono più linee telefoniche intelligenti che bambini. Le telecomunicazioni devono affrontare ogni volta maggiori sfide, impensabili solamente poco tempo fa, come l'aumento del volume dei dati trasmessi (per esempio, oggi possiamo vedere la televisione sui dispositivi mobili, impensabile fino a 5 anni fa), migliori sistemi di cifratura per affrontare un ambiente ostile o nuove formule per trasmettere segnali che permettano un maggiore miniaturizzazione. La matematica ha la risposta a tutte queste sfide, e tutto ciò che passa, in larga misura, per l'uso della teoria dell'informazione.
I principali apporti alla teoria dell'informazione provengono praticamente quasi tutti dal matematico Claude E. Shannon, che non solo stabilì le basi teoriche della teoria dell'informazione, ma lavorò anche intensamente allo sviluppo di codici ottimizzati per trasmettere l'informazione. Tutto il primo capitolo è dedicato all'entropia di Shannon, forse l'invenzione più famosa e lo strumento più potente lasciatoci dal *padre della teoria dell"informazione*.
Tuttavia, i progressi in solitaria di Shannon in questa disciplina a metà del XX secolo sono stati ampliati e rapidamente superati dall'avvento dell'Era delle Comunicazioni. Con la nascita di Internet, le comunicazioni tra computer, hanno dovuto necessariamente svilupparsi a tappe forzate, per passare dai rudimentali metodi su cui si basavano le comunicazioni analogiche (telegrafo, telefono, ecc.) alla comunicazione digitale come la conosciamo oggi. Il secondo capitolo è completamente dedicato alla creazione di un sistema di trasmissione dei dati o anche solo dei messaggi, che sia veloce e computazionalmente poco dispendioso. Si fa riferimento nella sezione, ai codici di Huffmann, a mio avviso semplicemente geniali, e alla trasformata di Burrows-Wheeler; quest'ultima è stata inserita perchè, a mio parere, cerca di risolvere il problema dell'efficienza di una trasmissione in modo più ingegneristico che matematico e ho voluto, in qualche modo, rendere onore anche a chi, di fatto, costruisce questi sistemi di comunicazione o molte altre cose, che i matematici si

limitano solo a teorizzare: gli ingegneri, appunto.
Dal terzo capitolo in poi si passa alla parte più pratica della trasmissione, ovvero la codifica, la decodifica e la eventuale correzione degli errori. Per fare tutto ciò si parte dalla nozione di distanza di Hamming, lo stesso Hamming che ha ideato un metodo di correzione così sensazionale da essere trattato in tutti i corsi elementari di algebra di tutti corsi triennali di matematica e non solo.
Una volta introdotto l'argomento in generale, si passa al quarto capitolo, dove vengono esposti i codici lineari, provvisti di proprietà che li rendono più facili da studiare, creando connessioni con l'algebra lineare.
Il quinto capitolo ha il compito introdurre le conoscenze per affrontare il successivo. Si occupa di esibire un metodo di controllo degli errori molto elementare, una rudimentale costruzione dei codici di Hamming e infine un metodo di correzione singola e rilevazione doppia. Unisce semplicemente i due metodi presentati nelle sezioni 5.1 e 5.2.
Per finire, si passa al sesto ed ultimo capitolo, del quale sono particolarmente fiero. Al suo interno espongo come si possa ragionare sui codici di Hamming vedendo i messaggi non come semplici sequenze di 1 e 0, ma come polinomi a coefficienti in un campo finito. Tale analogia mi ha sempre affascinato, perchè in qualche modo rispecchia esattamente ciò che io penso quando faccio matematica: un continuo intersecarsi delle branche, una perfetta sinergia di discipline, che si aiutano, grazie alla quale dove non arriva l'analisi può arrivare l'algebra o la geometria, e così via. Al suo interno, grazie all'aiuto del professor Lenci, ho voluto provare ad inserire una visione più pratica di tutto l'argomento della correzione, per provare a determinare quali codici con quali proprietà vanno usati in date circostanze, sapute le caratteristiche del canale di comunicazione. Il capitolo si conclude con una formula generale che racchiude tutto i concetti affrontati, per la costruzione di un codice che può correggere un numero arbitrario di errori usando un campo finito qualsiasi.

1 Entropia di Shannon

Definizione 1.1. Dato uno spazio di probabilità $(\Omega, \mathcal{A}, \mathbb{P})$ e una partizione $\mathscr{P} = \{P_i\}_{i=1}^n$ si definisce l'*entropia* della partizione

$$H[\mathscr{P}] := -\sum_{i=1}^{n} \mathbb{P}(P_i) \log_2(\mathbb{P}(P_i))$$

prendendo come convenzione $0 \cdot \log_2 0 = 0$
Da qui in seguito, se non specificato, quando sarà scritto "log" si intenderà sempre "$\log_2$"

Definizione 1.2. Due variabili aleatorie $X, Y : \Omega \to \mathbb{R}$ sono *indipendenti* se e solo se

$$\mathbb{P}(X^{-1}(A) \cap Y^{-1}(B)) = \mathbb{P}(X^{-1}(A))\mathbb{P}(Y^{-1}(B)) \qquad \forall A, B \in \mathscr{B}(\mathbb{R})$$

Definizione 1.3. Un altro modo per scrivere l'entropia è

$$H[\mathscr{P}] = \sum_{i=1}^{n} z(\mathbb{P}(P_i)) \qquad \text{con } z(x) = \begin{cases} 0 & \text{se } x = 0 \\ -x \log x & \text{se } x \in (0, 1] \end{cases} \tag{1}$$

Osservazione. Sia $\mathscr{N} = \{\Omega\}$ la partizione banale. Allora

$$H[\mathscr{N}] = \sum_{i=1}^{1} \mathbb{P}(\Omega) \log \mathbb{P}(\Omega) = 1 \cdot \log(1) = 0$$

Osservazione. Sia $\mathscr{P}$ una partizione uniforme di n elementi, ovvero $\mathbb{P}(P_i) = \frac{1}{n}$ $\forall i$. Allora

$$H[\mathscr{P}] = -\sum_{i=1}^{n} \frac{1}{n} \log \frac{1}{n} = n \cdot \frac{1}{n} \log n = \log n$$

Proposizione 1.4. *Se $|\mathscr{P}| = n$ allora*

$$H[\mathscr{P}] \leq \log n$$

L'uguaglianza vale se e solo se $\mathscr{P}$ è una partizione uniforme.

Definizione 1.5. Siano due partizioni $\mathscr{P} = \{P_i\}_{i=1}^n$ e $\mathscr{Q} = \{Q_j\}_{j=1}^m$. Si dice che $\mathscr{Q}$ è più fine di $\mathscr{P}$ (è un raffinamento di $\mathscr{P}$) : $\mathscr{P} \leq \mathscr{Q}$ se e solo se

$$\forall Q_j \in \mathscr{Q} \; \exists \; P_i \in \mathscr{P} \text{ tc } Q_j \subset P_j \mod \mathbb{P} \quad \text{(a meno di insiemi di misura nulla)}$$

Definizione 1.6. Date due partizioni $\mathscr{P}$ e $\mathscr{Q}$, si chiama *inviluppo* di $\mathscr{P}$ e $\mathscr{Q}$ $\mathscr{P} \vee \mathscr{Q}$, la partizione più grossolana più fine sia di $\mathscr{P}$ che di $\mathscr{Q}$.

Definizione 1.7. Date due partizioni $\mathscr{P}$ e $\mathscr{Q}$, si chiama *intersezione* di $\mathscr{P}$ e $\mathscr{Q}$ $\mathscr{P} \wedge \mathscr{Q}$, la partizione più fine per cui sia $\mathscr{P}$ che $\mathscr{Q}$ sono raffinamenti.

Definizione 1.8. Data $\mathscr{P}$ partizione e $B \subset \Omega$ si definisce la partizione $\mathscr{P}_B$ indotta da $\mathscr{P}$ su B come

$$\{P_i \cap B\}_{i=1}^n$$

Se $\mathbb{P}(B) > 0$, vale

$$\mathbb{P}_B(\cdot) = \mathbb{P}(\cdot|B) = \frac{\mathbb{P}(\cdot \cap B)}{\mathbb{P}(B)}.$$

$\mathbb{P}_B$ è una misura di probabilità su B. Tale misura rende lo spazio misurabile $(B, \mathcal{A}_B)$, con $\mathcal{A}_B = \{A \cap B | A \in \mathcal{A}\}$ uno spazio di probabilità.

Definizione 1.9. Si definisce *entropia condizionale* di $\mathscr{P}$ rispetto a $\mathscr{Q}$ la quantità

$$H[\mathscr{P}|\mathscr{Q}] = \sum_{j=1}^{m} \mathbb{P}(Q_j) \sum_{i=1}^{n} z(\mathbb{P}(P_i|Q_j)) = \sum_{j=1}^{m} \mathbb{P}(Q_j) H[\mathscr{P}_{Q_j}]$$

Osservazione. 1.

$$H[\mathscr{P}|\mathscr{N}] = \sum_{j=1}^{1} \mathbb{P}(\Omega) H[\mathscr{P}_\Omega] = \mathbb{P}(\Omega) H[\{P_i \cap \Omega\}_{i=1}^n] = H[\{P_i\}_{i=1}^n] = H[\mathscr{P}]$$

2. $\mathscr{P} \vee \mathscr{N} = \mathscr{P}$ poichè la partizione più grossolana che è raffinamento sia di $\mathscr{P}$ che di $\mathscr{N}$ è proprio $\mathscr{P}$.

Si da ora una definizione e un lemma utile per un teorema seguente.

Definizione 1.10. Data $f : \mathbb{R}^n \to \mathbb{R}$, si dice *convessa* [*concava*] se

$$\forall x_1, x_2, \forall \lambda_1, \lambda_2 \in [0,1] \text{ tc } \lambda_1 + \lambda_2 = 1$$

vale

$$f(\lambda_1 x_1 + \lambda_2 x_2) \leq \lambda_1 f(x_1) + \lambda_2 f(x_2)$$
$$[\geq]$$

E' detta *strettamente convessa* [*strettamente concava*] se è convessa [concava] e

$$f(\lambda_1 x_1 + \lambda_2 x_2) = \lambda_1 f(x_1) + \lambda_2 f(x_2) \Rightarrow x_1 = x_2 \vee \lambda_1 \lambda_2 = 0$$

Lemma 1.11. *(Jensen) Sia $X : \Omega \to \mathbb{R}^n$ una variabile aleatoria su $(\Omega, \mathcal{A}, \mathbb{P})$ e $f : \mathbb{R}^n \to \mathbb{R}$ convessa [concava], allora*

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$$
$$[\geq]$$

Inoltre, se f è strettamente convessa [strettamente concava] allora

$$f(\mathbb{E}[X]) = \mathbb{E}[f(X)] \Rightarrow X = costante$$

Dimostrazione. In questo caso, è sufficiente dimostrare il lemma nel caso di Ω discreto. Dato $\Omega = \{\omega_i\}_{i=1}^n$, si procede per induzione completa su n.
Sia $\Omega = \{\omega_1, \omega_2\}$ e siano $p_i = \mathbb{P}(\omega_i)$ e $x_i = X(\omega_i)$. La disuguaglianza diventa

$$f(p_1 x_1 + p_2 x_2) \leq p_1 f(x_1) + p_2 f(x_2),$$

che segue direttamente dalla definizione di funzione convessa.
Sia vero per $\Omega = \{\omega_1, ...\omega_{n-1}\}$. Si indichi con $p_i' = p_i/(1 - p_n)$ per $i = 1, 2, ..., n-1$.

$$\sum_{i=1}^{n} p_i f(x_i) = p_n f(x_n) + (1 - p_n) \sum_{i=1}^{n-1} p_i' f(x_i) \geq$$

$$\geq p_n f(x_n) + (1 - p_n) f\left(\sum_{i=1}^{n-1} p_i' x_i\right) \geq f\left(p_n x_n + (1 - p_n) \sum_{i=1}^{n-1} p_i' x_i\right) = \qquad (2)$$
$$= f\left(\sum_{i=1}^{n} p_i x_i\right)$$

dove la disuguaglianza (2) segue dalla definizione di convessità. □

Teorema 1.12. *Siano $\mathscr{P} = \{P_i\}_{i=1}^n, \mathscr{Q} = \{Q_i\}_{j=1}^m, \mathscr{R} = \{R_i\}_{k=1}^t$ partizioni. Allora*

1. $H[\mathscr{P}|\mathscr{Q}] \geq 0 \qquad (= 0 \iff \mathscr{P} \leq \mathscr{Q})$
2. $H[\mathscr{P} \vee \mathscr{Q}|\mathscr{R}] = H[\mathscr{P}|\mathscr{R}] + H[\mathscr{Q}|\mathscr{P} \vee \mathscr{R}]$
3. $\mathscr{P} \leq \mathscr{Q} \Rightarrow H[\mathscr{P}|\mathscr{R}] \leq H[\mathscr{Q}|\mathscr{R}]$ *(l'entropia condizionale è monotona crescente nella prima variabile)*
4. $\mathscr{Q} \leq \mathscr{R} \Rightarrow H[\mathscr{P}|\mathscr{Q}] \leq H[\mathscr{P}|\mathscr{R}]$ *(l'entropia condizionale è monotona decrescente nella seconda variabile)*

5. $H[\mathscr{P} \vee \mathscr{Q}|\mathscr{R}] \leq H[\mathscr{P}|\mathscr{R}] + H[\mathscr{Q}|\mathscr{R}]$

Dimostrazione. 1. L'entropia, anche condizionale, per come è definita, è sempre positiva.

$$H(\mathscr{P}|\mathscr{Q}) = 0 \iff \forall j \ \mathscr{P}_{Q_j} = \mathscr{N}_{Q_j} \mod \mathbb{P} \text{ (partizione banale)}$$

Che è equivalente a dire che la partizione $\mathscr{P}$ non spezza l'insieme Q_j.

$\forall j \ \mathscr{P}_{Q_j} = \mathscr{N}_{Q_j} \mod \mathbb{P}$ (partizione banale) $\iff \exists\, i \ \text{tc} \ Q_i \subseteq P_i \mod \mathbb{P} \iff \mathscr{P} \leq \mathscr{Q}$

2.

$$H[\mathscr{P} \vee \mathscr{Q}|\mathscr{R}] = -\sum_k \mathbb{P}(R_k) \sum_{i,j} z(\mathbb{P}(P_i \cap Q_j|R_k)) =$$

$$-\sum_{i,j,k} \mathbb{P}(R_k) \frac{\mathbb{P}(P_i \cap Q_j \cap R_k)}{\mathbb{P}(R_k)} \log \frac{\mathbb{P}(P_i \cap Q_j \cap R_k)}{\mathbb{P}(R_k)} = \tag{3}$$

$$\left(\frac{\mathbb{P}(P_i \cap Q_j \cap R_k)}{\mathbb{P}(R_k)} = \frac{\mathbb{P}(P_i \cap R_k)}{\mathbb{P}(R_k)} \cdot \frac{\mathbb{P}(P_i \cap Q_j \cap R_k)}{\mathbb{P}(P_i \cap R_k)} = \mathbb{P}(P_i|R_k) \cdot \mathbb{P}(Q_j|P_i \cap R_k) \right)$$

Il passaggio è lecito: i valori di $\mathbb{P}(P_i \cap R_k)$ che uguali a zero, per qualche coppia i, k sono stati già esclusi dalla sommatoria in (3).

$$= -\sum_{i,j,k} \mathbb{P}(P_i \cap Q_j \cap R_k) \cdot \log \mathbb{P}(P_i|R_k) - \sum_{i,j,k} \mathbb{P}(P_i \cap Q_j \cap R_k) \cdot \log \mathbb{P}(Q_j|P_i \cap R_k) =$$

$$= -\sum_{i,j,k} \mathbb{P}(P_i \cap R_k) \cdot \mathbb{P}(Q_j|P_i \cap R_k) \cdot \log \mathbb{P}(P_i|R_k) -$$

$$+ \sum_{i,j,k} \mathbb{P}(P_i \cap R_k) \cdot \mathbb{P}(Q_j|P_i \cap R_k) \cdot \log \mathbb{P}(Q_j|P_i \cap R_k) =$$

$$= -\sum_{i,k} \mathbb{P}(P_i \cap R_k) \cdot \log \mathbb{P}(P_i|R_k) \cdot \sum_j \mathbb{P}(Q_j|P_i \cap R_k) -$$

$$+ \sum_{i,k} \mathbb{P}(P_i \cap R_k) \cdot \sum_j \mathbb{P}(Q_j|P_i \cap R_k) \cdot \log \mathbb{P}(Q_j|P_i \cap R_k) =$$

$$= -\sum_{i,k} \mathbb{P}(R_k)\mathbb{P}(P_i|R_k) \cdot \log \mathbb{P}(P_i|R_k) + \sum_{i,k} \mathbb{P}(P_i \cap R_k) \cdot \sum_j z(\mathbb{P}(Q_j|P_i \cap R_k)) =$$

$$= \sum_k \mathbb{P}(R_k) \sum_i z(\mathbb{P}(P_i|R_k)) + H[\mathscr{Q}|\mathscr{P} \vee \mathscr{R}] =$$

$$= H[\mathscr{P}|\mathscr{R}] + H[\mathscr{Q}|\mathscr{P} \vee \mathscr{R}] =$$

3. Se $\mathscr{P} \leq \mathscr{Q}$ si ha $\mathscr{P} \vee \mathscr{Q} = \mathscr{Q}$ quindi per 2.

$$H[\mathscr{Q}|\mathscr{R}] = H[\mathscr{P} \vee \mathscr{Q}|\mathscr{R}] = H[\mathscr{P}|\mathscr{R}] + H[\mathscr{Q}|\mathscr{P} \vee \mathscr{R}] \geq H[\mathscr{P}|\mathscr{R}]$$

4. Si sfrutta la disuguaglianza (1.11) per z come in 1 concava

$$\sum_k \mathbb{P}(R_k|Q_j) z(\mathbb{P}(P_i|R_k)) \leq z\left(\sum_k \mathbb{P}(R_k|Q_j)\mathbb{P}(P_i|R_k)\right) \tag{4}$$

Infatti $\mathbb{P} \geq 0$ e $\sum_k \mathbb{P}(R_k|Q_j) = 1 \ \forall j$. Dato che $\mathscr{Q} \leq \mathscr{R}$ vale che $\forall j \ Q_j = \dot{\bigcup}_{k_j} R_{k_j}$. Perciò

$$\sum_k \mathbb{P}(R_k|Q_j)\mathbb{P}(P_i|R_k) = \sum_{k_j} \frac{\mathbb{P}(P_i \cap R_{k_j})}{\mathbb{P}(R_{k_j})} \frac{\mathbb{P}(R_{k_j})}{\mathbb{P}(Q_j)} =$$

Per i $k \neq k_j$ la sommatoria è nulla: $R_k \cap Q_j = \begin{cases} R_k & \text{per } k = k_j \\ \varnothing & \text{per } k \neq k_j \end{cases}$

$$= \sum_{k_j} \frac{\mathbb{P}(P_i \cap R_{k_j})}{\mathbb{P}(Q_j)} = \frac{\mathbb{P}(P_i \cap Q_j)}{\mathbb{P}(Q_j)} = \mathbb{P}(P_i|Q_j)$$

Ora si sostituisce in (4) moltiplicando entrambi i membri per $\mathbb{P}(Q_j)$ e si somma su j e i.

$$\sum_{i,j,k} \mathbb{P}(R_k|Q_j)\mathbb{P}(Q_j) z(\mathbb{P}(P_i|R_k)) \leq \sum_{i,j} \mathbb{P}(Q_j) z(\mathbb{P}(P_i|Q_j)) \Rightarrow$$

$$\Rightarrow \sum_{j,k} \mathbb{P}(R_k \cap Q_j) \sum_i z(\mathbb{P}(P_i|R_k)) \leq \sum_j, \mathbb{P}(Q_j) \sum_i z(\mathbb{P}(P_i|Q_j)) \Rightarrow$$

$$\Rightarrow \sum_k \mathbb{P}(R_k) \sum_i z(\mathbb{P}(P_i|R_k)) \leq \sum_j, \mathbb{P}(Q_j) \sum_i z(\mathbb{P}(P_i|Q_j)) \Rightarrow H[\mathscr{P}|\mathscr{R}] \leq H[\mathscr{P}|\mathscr{Q}]$$

5. Dal punto 2 è sufficiente dimostrare che $H[\mathscr{Q}|\mathscr{R}] \geq H[\mathscr{Q}|\mathscr{P} \vee \mathscr{R}]$, ma poichè $\mathscr{P} \vee \mathscr{R} \geq \mathscr{R}$, banalmente, per il punto 4. è immediato. □

Corollario 1.13.

2'. $H[\mathscr{P} \vee \mathscr{Q}] = H[\mathscr{P}] + H[\mathscr{Q}|\mathscr{P}]$

3'. $\mathscr{P} \leq \mathscr{Q} \Rightarrow H[\mathscr{P}] \leq H[\mathscr{Q}]$

4'. $H[\mathscr{P}|\mathscr{Q}] \leq H[\mathscr{P}]$

5'. $H[\mathscr{P} \vee \mathscr{Q}] \leq H[\mathscr{P}] + H[\mathscr{Q}]$

Dimostrazione. Si noti che vale $H[\mathscr{P}] = H[\mathscr{P}|\mathscr{N}]$.
2'. Deriva dal punto 2. ponendo $\mathscr{R} = \mathscr{N}$
3'. Deriva dal punto 3. ponendo $\mathscr{R} = \mathscr{N}$
4'. Deriva dal punto 4. ponendo $\mathscr{Q} = \mathscr{N}$ e $\mathscr{R} = \mathscr{Q}$
5'. Deriva dal punto 5. ponendo $\mathscr{R} = \mathscr{N}$ □

1.1 Entropia di variabili aleatorie

Definizione 1.14. Si X variabili aleatorie. Questa si dice *finita* se $Im(X) < \infty$.

Siano X, Y due variabili aleatorie finite, sullo spazio di probabilità $(\Omega, \mathcal{A}, \mathbb{P})$. In generale, si chiamerà $\mathcal{X} = Im(X)$ e $\mathcal{Y} = Im(Y)$.
Ad ogni variabile aleatoria X corrisponde una partizione di Ω, che verrà chiamata $\mathscr{P}_X$, tale per cui

$$\mathscr{P} = \{X^{-1}(x)\}_{x \in \mathcal{X}}$$

Definizione 1.15. Date X, Y variabile aleatoria e $\mathscr{P}_X, \mathscr{P}_Y$ partizioni relative rispettivamente ad X, Y, come sopra, allora

- $H[X] = H[\mathscr{P}_X]$
- $H[X|Y] = H[\mathscr{P}_X|\mathscr{P}_Y]$
- $H[X,Y] = H[\mathscr{P}_X \vee \mathscr{P}_Y]$ (entropia congiunta)

$H[X,Y]$ è semplicemente $H[Z]$ dove $Z := (X,Y)$; infatti

$$\mathscr{P}_Z = \{Z^{-1}(x,y)\}_{(x,y)\in(\mathcal{X},\mathcal{Y})} = \{\omega \in \Omega | Z(\omega) = (X(\omega), Y(\omega)) =$$
$$= (x,y)\}_{(x,y)\in(\mathcal{X},\mathcal{Y})} = \{\omega \in \Omega | X(\omega) = x, Y(\omega) = y\}_{(x,y)\in(\mathcal{X},\mathcal{Y})} =$$
$$= \{X^{-1}(x) \cap Y^{-1}(y)\}_{(x,y)\in(\mathcal{X},\mathcal{Y})} = \mathscr{P}_X \vee \mathscr{P}_Y$$

Osservazione. Siccome $H[X]$ dipende solo da $\mathscr{P}_X$, allora tutte le variabili aleatorie Y tali che $\mathscr{P}_Y = \mathscr{P}_X$ hanno la stessa entropia, ovvero $H[X] = H[Y]$.

Osservazione. L'entropia di Shannon dice l'informazione di un esperimento che si possiede, conoscendo solamente la variabile X.
L'entropia $H[Y,X]$ dice informazioni in cui fornisce la variabile Y, conoscendo le informazioni fornite dalla variabile X.
L'entropia $H[X,Y]$ dice informazioni che si hanno guardando contemporaneamente i due strumenti di misura X e Y. In sostanza, H[X,Y] dice l'informazione che si ha guardando prima lo srumento X e poi anche lo strumento Y.

Proposizione 1.16. *Siano $X_1, ..., X_n, Y$ variabili aleatorie; allora*

$$H[X_1, ..., X_n|Y] = \sum_{i=1}^{n} H[X_i|X_{i-1}, ..., X_1, Y]$$

Dimostrazione. Si dimostra per induzione su n
Sia $n = 2$

$$H[X_1, X_2|Y] = H[X_1|Y] + H[X_2|X_1, Y]$$

Per il punto 2. di 1.12
Sia valido per $n - 1$:

$$H[X_1, ..., X_n|Y] = H[X_1|Y] + H[X_2, ..., X_n|X_1, Y] =$$

$$= H[X_1|Y] + \sum_{i=2}^{n} H[X_i|X_{i-1}, ..., X_1, Y] = H[X_1, ..., X_n|Y]$$

□

Corollario 1.17.

$$H[X_1, ..., X_n] = \sum_{i=1}^{n} H[X_i|X_{i-1}, ..., X_1]$$

Dimostrazione. Si applica il teorema precedente con $Y = costante$ □

Proposizione 1.18. *(Principio del pettegolezzo) Sia $X : \Omega \to \mathcal{X}$ suriettiva, $\phi : \mathcal{X} \to \mathcal{Y}$, anch'essa suriettiva. Si ponga $Y = \phi \circ X : \Omega \to \mathcal{Y}$. Allora $H[Y] \leq H[X]$, e vale l'uguaglianza se e solo se ϕ è biettiva.*

Dimostrazione. Basta dimostrare che $\mathscr{P}_Y \leq \mathscr{P}_X$, si conclude poi con il punto 3'. di 1.13. Sia $\omega \in X^{-1}(\mathcal{X})$. Allora

$$\omega \in X^{-1}(\mathcal{X}) \Rightarrow \omega \in X^{-1}(\phi^{-1}(\phi(\mathcal{X}))) \Rightarrow \omega \in Y^{-1}(\phi(\mathcal{X}) \Rightarrow X^{-1}(\mathcal{X}) \subseteq Y^{-1}(\phi(\mathcal{X})).$$

Si mostra ora l'uguaglianza.
($\Leftarrow$) ϕ biettiva $\Rightarrow \mathscr{P}_X = \mathscr{P}_y$. Infatti

$$\mathscr{P}_X = \{\{\omega \in \Omega | X(\omega) = x_i\}\}_{i=1}^{n} = \{\{\omega \in \Omega | \phi(X(\omega)) = \phi(x_i)\}\}_{i=1}^{n} =$$

$$= \{\{\omega \in \Omega | Y(\omega) = y_i\}\}_{i=1}^{n} = \mathscr{P}_Y \Rightarrow H[X] = H[\mathscr{P}_X] = H[\mathscr{P}_Y] = H[Y]$$

($\Rightarrow$) Per la prima parte della proposizione vale $\mathscr{P}_Y \subseteq \mathscr{P}_X (\Rightarrow H[Y] \leq H[X])$ e per ipotesi $H[X] = H[Y]$.
Poichè $\mathscr{P}_Y \leq \mathscr{P}_X$ allora $\mathscr{P}_X \vee \mathscr{P}_Y = \mathscr{P}_X = H[X,Y] = H[X]$. Dunque per 1.16

$$H[X,Y] = H[X] = H[Y] + H[X|Y]$$

Si ottiene

$$H[X] = H[Y] + H[X|Y] \Rightarrow H[X|Y] = 0 \Rightarrow \mathscr{P}_Y \geq \mathscr{P}_X \text{ per il primo punto di 1.12}$$

Se ϕ non fosse biettiva, esisterebbero due indici i, j tali che $x_i \neq x_j$ per cui $\phi(x_i) = \phi(x_j) = y$ e $X^{-1}(x_i) \cap X^{-1}(x_j) = \varnothing$. Ma $Y^{-1}(y) \subseteq X^{-1}(x_i) \cup X^{-1}(x_j)$. Assurdo. Dunque ϕ è biettiva. □

1.2 Entropia relativa e mutua informazione

Osservazione. Per alleggerire la notazione usata, d'ora in poi varranno le seguenti:

- $p(x) := \mathbb{P}(X = x) \qquad (x \in Im(X) = \mathcal{X})$
- $p(x,y) = \mathbb{P}(X = x, Y = y)$
- $p(x|y) = \mathbb{P}(X = x | Y = y)$

In $p(\cdot,:)$ il primo valore si riferisce sempre alla variabile X, il secondo sempre alla variabile Y. Perciò $p(y,x)$ si riferisce a $\mathbb{P}(X = y, Y = x)$ e $p(x,x)$ è $\mathbb{P}(X = x, Y = x)$. Altri esempi sono

$$H[X] = -\sum_x p(x) \log p(x) = \mathbb{E}_p[\log p(X)], \; H[X|Y] =$$

$$= -\sum_y p(y) \sum_x p(x|y) \log p(x|y)$$

Definizione 1.19. Dato Ω spazio di probabilità, $X : \Omega \to \mathcal{X}$ suriettiva. Si consideri un'altra misura di probabilità $\mathbb{Q}$ su Ω e si chiami

$$q(x) = \mathbb{Q}(\{X^{-1}(x)\})$$

Si chiama *entropia relativa* o *distanza di Kullback - Leibler* fra le distribuzioni $p = \{p(x)\}_{x \in \mathcal{X}}$ e $q = \{q(x)\}_{x \in \mathcal{X}}$ la quantità

$$D(p||q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}$$

Per convenzione sarà

$$0 \cdot \log \frac{0}{q} = 0 \text{ se } q > 0 \quad \text{e} \quad 0 \cdot \log \frac{p}{0} = +\infty \text{ se } p > 0$$

Proprietà:

1. $D(p||q) > 0$, l'uguaglianza vale se e solo se $p = q$
2. Non è necessariamente simmetrica
3. Non necessariamente vale la disuguaglianza triangolare

Dimostrazione. 1. Definisco la variabile aleatoria Y che vale $Y(\omega) := \frac{q(x)}{p(x)}$ su ciascun insieme $X^{-1}(x) = \{\omega \in \Omega | X(\omega) = x\}$.
Quindi $-D(p||q) = \mathbb{E}_p(\log Y)$. Poichè la funzione log è una funzione concava, per 1.11 si ha

$$\mathbb{E}_p(\log Y) \leq \log(\mathbb{E}_p[Y]) = \log\left(\sum_x p(x)\frac{q(x)}{p(x)}\right) = \log\left(\sum_x q(x)\right) = \log(1) = 0$$

Per la seconda parte di 1.11, poichè log è strettamente concava:

$$D(p||q) = 0 \iff Y = c \text{ costante } \iff \forall\ x\ \frac{q(x)}{p(x)} = c \iff$$

$$\iff \forall\ x\ p(x) = \frac{1}{c}q(x) \iff \sum_x p(x) = \sum_x \frac{1}{c}q(x) \iff$$

$$\iff \sum_x p(x) = \frac{1}{c}\sum_x q(x) \iff 1 = \frac{1}{c} \iff c = 1 \iff p \equiv q$$

□

Definizione 1.20. Date due variabili aleatorie X, Y, si definisce la *mutua informazione*

$$I(X;Y) = D(p(x|y)||p(x)(y)) = \sum_{x,y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}$$

Proposizione 1.21. *1.* $I(X;Y) = I(Y;X)$

2. $X = Y \Rightarrow I(X;X) = H[X]$

3. $I(X;Y) = H[X] - H[X|Y]$

4. $I(X;Y) = H[X] + H[Y] - H[X,Y]$

5. $0 \leq I(X;Y) \leq \max\{H[X], H[Y]\}$

Dimostrazione. 1. Segue dalla definizione
2.

$$I(X;Y) = I(X;X) = \sum_x p(x) \log\left(\frac{p(x)}{p(x)p(x)}\right) = \sum_x p(x) \log\left(\frac{1}{p(x)}\right) = H[X]$$

3.

$$I(X;Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} = \sum_{x,y} p(x,y) \log \frac{p(x|y)}{p(x)} =$$

$$= \sum_{x,y} p(x,y) \log p(x|y) - \sum_{x,y} p(x,y) \log p(x) =$$

$$= \sum_{x,y} p(y)p(x|y) \log p(x|y) - \sum_x p(x) \log p(x) \sum_y p(y|x) =$$

$$= -\left(-\sum_y p(y) \sum_x p(x|y) \log p(x|y)\right) - \sum_x p(x) \log p(x) = H[X] - H[X|Y]$$

4. Per 1.16 $H[X|Y] = H[X,Y] - H[Y]$.
Per 3.

$$I(X;Y) = H[X] - H[X|Y] = H[X] + H[Y] - H[X,Y]$$

5. $I(X;Y) \geq 0$ per definizione di mutua informazione.
Per i punto 3. $I(X;Y) \leq H[X]$ e simmetricamente $I(X;Y) \leq H[Y]$.
Perciò, $I(X;Y) \leq \max\{H[X], H[Y]\}$ □

Proposizione 1.22. *Siano X,Y variabili aleatorie. Si ha che*

$$H[X|Y] \leq H[X]$$

Con l'uguaglianza se e solo se X e Y sono indipendenti.

Dimostrazione. La disuguaglianza vale per il punto 3. della precedente proposizione:

$$0 \leq I(X;Y) = H[X] - H[X|Y] \Rightarrow H[X|Y] \leq H[X]$$

$$X, Y \text{ indipendenti } \iff p(x,y) = p(x)p(y) \iff I(X;Y) = 0 \Leftrightarrow H[X|Y] = H[X]$$

□

Definizione 1.23. Si definisce la *probabilità uniforme* come $u(x) := \frac{1}{|\mathcal{X}|}$

Proposizione 1.24. *Data X variabile aleatoria, si ha che*

$$H[X] \leq \log |\mathcal{X}|$$

Con l'uguaglianza se e solo se $p(x) = u(x)$

Dimostrazione. Si consideri

$$0 \leq D(p||u) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{u(x)} = \sum_{x \in \mathcal{X}} p(x) \log p(x) + \sum_{x \in \mathcal{X}} p(x) \log |\mathcal{X}| =$$

$$= - \left(- \sum_{x \in \mathcal{X}} p(x) \log p(x) \right) + \log |\mathcal{X}| \sum_{x \in \mathcal{X}} p(x) = \log |\mathcal{X}| - H[X] \iff H[X] \leq \log |\mathcal{X}|$$

$$p \equiv u \iff D(p||u) = 0 \iff H[X] = \log |\mathcal{X}|$$

□

Esempio 1.25. Si consideri la variabile aleatoria X con $\mathcal{X} = \{1, 2, 3, 4\}$, dove ogni evento ha probabilità descritte in tabella.
Si supponga per assurdo che la variabile X abbia distribuzione uniforme $u(x)$; si codifichi l'informazione di ciascuna uscita di X secondo la procedura.

X	p(x)	codice p
1	1/2	0
2	1/4	10
3	1/8	110
4	1/8	111

X	u(x)	codice u
1	1/4	00
2	1/4	01
3	1/4	10
4	1/4	11

Il consumo medio di bit per ogni uscita di X secondo il codice u è 2.

$$\mathbb{E}[l(X)] = \frac{1}{4} \cdot 2 + \frac{1}{4} \cdot 2 + \frac{1}{4} \cdot 2 + \frac{1}{4} \cdot 2 = 2$$

Con la distribuzione p, il codice migliore avrebbe comportato un consumo medio per uscita di $\frac{7}{4}$ di bit

$$\mathbb{E}[l(X)] = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \frac{1}{8} \cdot 3 = 1,75$$

La differenza di consumo medio e dunque lo spreco, è $2 - 1,75 = 0,25 = 1/4$
Si osserva che

$$D(p||q) = \log(|\mathcal{X}|) - H[X] = 2 - \frac{7}{4} = \frac{1}{4}$$

Si vedrà in seguito ce questo fatto è generale, se si usa un codice sbagliato (cioè pensato per adattarsi alla distribuzione uniforme u quando la reale distribuzione è p): si aggiunge sempre uno spreco di bit per uscita uguale a $D(p||q)$, dove p è la distribuzione corretta e q la distribuzione supposta.

Definizione 1.26. Si definisce *mutua informazione condizionata*, date X, Y, Z variabili aleatorie

$$I(X;Y|Z) := H[X|Z] - H[X|Y,Z]$$

Osservazione. Se Z è una costante, quindi corrispondente alla partizione banale, si ottiene $I(X:Y|Z) = I(X:Y)$

Proposizione 1.27.

$$I(X_1,...,X_n;Y) = \sum_{i=1}^{n} I(X_i;Y|X_{i-1},...,X_1)$$

Dimostrazione. La dimostrazione discende dalla definizione di mutua informazione condizionata e dal corollario 1.17

$$I(X_1,...,X_n;Y) = H[X_1,...,X_n] - H[X_1,...,X_n|Y] =$$

$$= \sum_{i=1}^{n} H[X_i|X_{i-1},...,X_1] - \sum_{i=1}^{n} H[X_i|X_{i-1},...,X_1,Y] =$$

$$= \sum_{i=1}^{n} (H[X_i|X_{i-1},...,X_1] - H[X_i|X_{i-1},...,X_1,Y]) =$$

$$= (H[X_1] - H[X_1|Y]) + (H[X_2|X_1] - H[X_2|Y,X_1]) + ...$$

$$... + (H[X_n|X_{n-1},...,X_1] - H[X_n|Y,X_{n-1},...,X_1]) =$$

$$= I(X_1;Y) + I(X_2;Y|X_1) + ... + I(X_n;Y|X_{n-1},...,X_1) =$$

$$= \sum_{i=1}^{n} I(X_i;Y|X_{i-1},...,X_1)$$

□

Proposizione 1.28. *(Stima dell'entropia congiunta)*

$$H[X_1,...,X_n] \leq \sum_{i=1}^{n} H[X_i]$$

Dimostrazione. La dimostrazione vale per 1.17 e per 1.22

$$H[X_1,...,X_n] = \sum_{i=1}^{n} H[X_i|X_{i-1},...,X_1] =$$

$$H[X_1] + H[X_2|X_1] + ... + H[X_n|X_{n-1},...,X_1] \leq$$

$$\leq H[X_1] + H[X_2] + ... + H[X_n|X_{n-1}, ..., X_1] \leq ...$$

$$... \leq H[X_1] + H[X_2] + ... + H[X_n] = \sum_{i=1}^{n} H[X_i]$$

□

Colui che guarda le informazioni da $X_1, ..., X_n$ ha meno informazioni (anche se ha le informazioni massime di tutti) della somma delle singole informazioni degli X_i poiché parte dell'informazione negli X_i potrebbe ripetersi.

Osservazione. Nella stima dell'entropia congiunta, vale l'uguaglianza se e solo se le variabili $X_1, ..., X_n$ son indipendenti.

Dimostrazione. ($\Rightarrow$) Ricordando 1.17, si procede per induzione su n. Per il passo base $n = 2$

$$H[X_1] + H[X_2|X_1] = H[X_1] + H[X_2] \Rightarrow H[X_2|X_1] = H[X_2] \Rightarrow$$

$$\Rightarrow X_1, X_2 \text{ sono indipendenti, per 1.22} \tag{5}$$

Sia valida ora per $n-1 : X_1, ..., X_{n-1}$ sono indipendenti

$$\sum_{i=1}^{n} H[X_i|X_{i-1}, ..., X_1] = \sum_{i=1}^{n} H[X_i] \Rightarrow$$

$$\Rightarrow \sum_{i=1}^{n-1} H[X_i|X_{i-1}, ..., X_1] + H[X_n|X_{n-1}, ..., X_1] = \sum_{i=1}^{n-1} H[X_i] + H[X_n] \Rightarrow$$

$$\Rightarrow H[X_n|X_{n-1}, ..., X_1] = H[X_n] \Rightarrow X_1, ..., X_n \text{ sono indipendenti per 1.22}$$

($\Leftarrow$) Se $X_1, ..., X_n$ sono indipendenti

$$\sum_{i=1}^{n} H[X_i] = \sum_{i=1}^{n} H[X_i|X_{i-1}, ..., X_1] = H[X_1, ..., X_n]$$

□

Per scrivere il capitolo appena esposto, sono stati consultati principalmente gli appunti del corso di Teoria dell'informazione e Comunicazione, tenuto dal professor Lenci, presso l'università di Bologna, agli studenti del Corso di Laurea Magistrale, Curriculum Generale Applicativo.

2 Compressione dati

Si supponga di avere a disposizione un *alfabeto* $\mathcal{D} = \{l_1, l_2, ...\}$ di simboli, che verranno chiamati *lettere*, che si possono trasmettere; si indica $\mathcal{D}^* := \bigcup_n \mathcal{D}^n$ l'insieme delle stringhe finite di elementi di $\mathcal{D}$, composte affiancando un numero finito di lettere.

Definizione 2.1. Un codice C per la variabile aleatoria $X : \Omega \to \mathcal{X}$, sull'alfabeto $\mathcal{D}$ è una funzione $C : \mathcal{X} \to \mathcal{D}^*$

Indicata con x la cifra "uscita" e $C(x)$ la parola di codice per la cifra x:

- se la cardinalità dell'alfabeto del codice è finita: $|\mathcal{D}| = D < \infty$, allora il codice viene detto D-ario.
- si indica con $l_C(x)$ o $l(x)$ la lunghezza di $C(x)$ in simboli di $\mathcal{D}$: $l_C(x)$ è l'unico intero n per cui $C(x) \in \mathcal{D}^n$

Definizione 2.2. Si definisce la *lunghezza media del codice* C come

$$L(C) = \sum_{x \in \mathcal{X}} p(x) l(x)$$

Se $\mathcal{D} = \{0, 1, ..., D-1\}$ con $D \in \mathbb{Z}^+$, ciascuno di questi simboli è chiamato D-it

2.1 Classificazione de codici

Definizione 2.3. Se C è una funzione iniettiva, allora il codice C si dice *non singolare*

Quando si trasmette il segnale ottenuto codificando $(X_i)_i$, esso avrà un aspetto del tipo

$$C(X_1)C(X_2)C(X_3)...$$

Le parole risulteranno trasmesse una di seguito all'altra e formeranno una stringa continua di dati; ovviamente, la scelta del codice C deve permettere di separare le singole parole, in modo che chi riceve il segnale lo possa decodificare. Tale operazione di divisione della stringa prende il nome di *parsing*.

Definizione 2.4. Si definisce $\mathcal{X}^*$ come l'insieme delle concatenazioni di un numero finito di elementi di $\mathcal{X}$.

Definizione 2.5. Si definisce $C^* : \mathcal{X}^* \to \mathcal{D}^*$, *estensione* di C nel seguente modo: dato un messaggio $x = x_1x_2...x_K$ allora

$$C^*(x) = C^*(x_1x_2...x_n) := C(x_1)C(x_2)...C(x_n)$$

Definizione 2.6. Un codice C viene detto *univocamente decodificabile* se C^* è non singolare.

Purtroppo anche i codici univocamente decodificabili possono presentare dei problemi dal punto di vista dell'utilizzo pratico

Esempio 2.7. Dato $\mathcal{X} = \{R, S, T, U\}$ si consideri il seguente codice binario univocamente decodificabile

$$C(R) = 10 \qquad C(S) = 00 \qquad C(T) = 11 \qquad C(U) = 110$$

Supponiamo ora di aver utilizzato l'estensione C per codificare un messaggio $x \in \mathcal{X}^+$ e di aver ottenuto la parola di codice 110...0. Durante il processo di decodifica, per capire se il primo simbolo del messaggio sorgente è T piuttosto che U si deve verificare se il numero di zeri che segue 11 sia pari o dispari. Infatti, se il numero di zeri è pari il messaggio dev'essere della forma $TS...S$. Se il numero di zeri è dispari il messaggio dev'essere della forma $US...S$

L'esempio mostra che un codice univocamente decodificabile può essere tale che per cominciare a decodificare il primo simbolo di un messaggio è necessario aspettare di leggere l'ultimo simbolo della codifica. Esaminando il codice, ci si accorge che il problema sta nel fato che la parola di codice per T è il prefisso della parola di codice U. Se nessuna parola di codice fosse prefisso di un'altra, sarebbe possibile decodificare i simboli sorgente mano a mano che viene ricevuto il messaggio.
Si noti che possiamo rimediare a questo problema riservando un simbolo di codice apposito per separare le codifiche dei simboli sorgente in una parola di codice Il codice dell'esempio sopra diventerebbe un codice ternario così composto

$$C(R) = 102 \qquad C(S) = 002 \qquad C(T) = 112 \qquad C(U) = 1102$$

Questa soluzione però compromette la compattezza del codice.
Un codice binario come il seguente

$$C(R) = 0 \qquad C(S) = 10 \qquad C(T) = 110 \qquad C(U) = 111$$

risolverebbe il problema

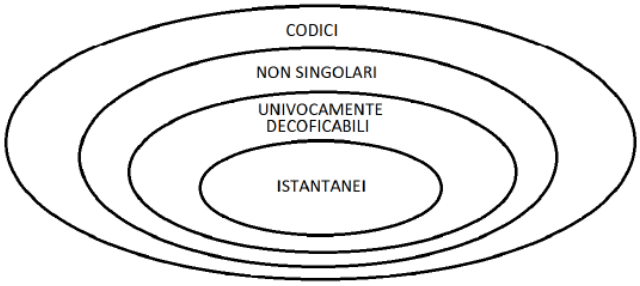


Figura 1: Classificazione dei vari codici

Definizione 2.8. Un codice è detto **istantaneo** (o *senza prefissi*) se nessuna parola di codice è prefisso di un'altra.

I codici istantanei soddisfano un'importate proprietà strutturale che li rende riconoscibili anche soltanto in base alle sole lunghezze delle parole di codice.

Teorema 2.9. ***(Disuguaglianza di Kraft)*** *Dati un codice istantaneo C sull'alfabeto $\mathcal{X}$ di cardinalità D e $l_1, ..., l_m \in \mathbb{Z}^+$ le lunghezze delle parole di codice, allora vale*

$$\sum_{i=1}^{m} D^{-l_i} \leq 1.$$

Viceversa, dato un insieme $\{l_1, ..., l_m\} \subset \mathbb{Z}^+$ che soddisfa la disuguaglianza sopra, allora esiste un codice istantaneo C tale che $l_1, ..., l_m$ sono le lunghezze delle parole di codice

Dimostrazione. (I parte) Sia l_{max} la lunghezza massima delle parole di c

$$l_{max} = \max_{i=1,...,m} l_C(x_i).$$

Si consideri l'albero D-ario completo, di profondità l_{max}. Si può posizionare ogni parola di codice di C su un nodo dell'albero seguendo dalla radice il cammino corrispondente ai simboli della parola. Dato che il codice è istantaneo, nessuna parola apparterrà al sottoalbero avente come radice un'altra parola. Possiamo quindi partizionare le foglie dell'albero in sottoinsiemi disgiunti $A_1, ..., A_m$ dove A_i è il sottoinsieme di foglie associato alla parola $c(x_i)$.

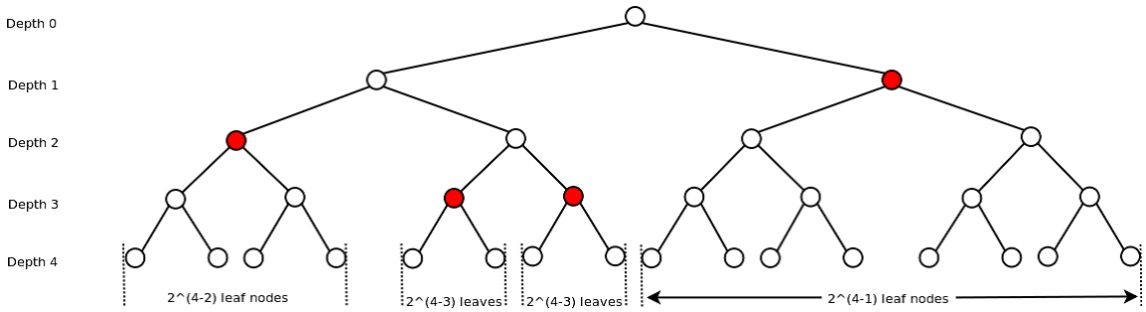


Figura 2: Partizione delle foglie indotta dal codice istantaneo indicato dai nodi colorati

Il numero di foglie nel sottoalbero di una parola ad altezza l_i è $D^{l_{max}-l_i}$. D'altra parte, il numero totale di foglie nell'albero è $D^{l_{max}}$. Quindi

$$\sum_{i=1}^{m} D^{l_{max}-l_i} = \sum_{i=1}^{m} |A_i| \leq D^{l_{max}}$$

Dividendo per $D^{l_{max}}$ si ottiene la disuguaglianza.
($\Leftarrow$) Si assuma che $l_1, ..., l_m > 0$ soddisfano la disuguaglianza di Kraft e sia $l_{max} = \max\{l_1, ..., l_m\}$. Allora si può costruire un codice istantaneo $C : \{x_1, ..., x_m\} \to \mathcal{D}^*$ con lunghezze date, ovvero $l(x_i) = l_i$ per $i = 1, ..., m$. A questo scopo si consideri l'albero D-ario ordinato e completo di profondità l_{max}. Al simbolo x_1 si associ la parola di codice $C(x_1)$ corrispondente al nodo dell'albero di altezza l_1 primo in ordine lessicografico (ovvero più a sinistra). Ad ogni simbolo successivo x_j si associ la parola di codice $C(x_j)$ corrispondente al primo nodo (sempre in ordine lessicografico) di altezza l_j che nè appartiene nè include sottoalberi radicati su parole scelte in precedenza. Si noti che il codice così costruito è istantaneo, dato che nessuna parola compare nel sottoalbero radicato su un'altra parola. Dato che le lunghezze soddisfano la disuguaglianza di Kraft, il numero totale di foglie necessarie a creare il codice è

$$\sum_{i=1}^{m} D^{l_{max}-l_i} \leq D^{l_{max}}$$

ovvero non maggiore delle foglie disponibili nell'albero. □

Definizione 2.10. La procedura così costruita per definire un codice istantaneo di lunghezze prefissate, viene detta *procedura di Shannon*.

2.2 Codici ottimali

Una comunicazione efficiente avviene quando la trasmissione di ogni simbolo di sorgente avviene in poco tempo, e quindi con pochi bit. Per avere mediamente un comportamento come quello richiesto, un codice di sorgente assegna le parole di lunghezza minore ai simboli di sorgente che hanno una più alta probabilità di emissione, e quelle di lunghezza maggiore ai simboli di sorgente che hanno una più bassa probabilità di emissione. E' vitale dunque minimizzare $L(C)$, la media delle lunghezze delle parole, con il vincolo della disuguaglianza di Kraft. Matematicamente, significa risolvere il seguente problema di minimo vincolato:

$$\begin{cases} L(C) = \sum_{i=1}^{m} p_i l_i \\ \sum_{i=1}^{m} D^{-l_i} \leq 1 \end{cases} \quad \text{per } l = (l_1, ..., l_m) \in (\mathbb{Z}^+)^m$$

Invece di minimizzare $f(l) = \sum_{i=1}^{m} p_i l_i$ su $(\mathbb{Z}^+)^m$, si cerca il minimo vincolato di f su $(\mathbb{R}^+)^m$. Il minimo trovato su $(\mathbb{R}^+)^m$ sarà minore o uguale al minimo su $(\mathbb{Z}^+)^m$. Senza perdita di generalità si può usare il vincolo bilatero $\sum_{i=1}^{m} D^{-l_i} = 1$, poiché se per caso il minimo fosse assunto per l_i^* tali che $\sum_{i=1}^{m} D^{-l_i^*} < 1$, allora si può trovare $\bar{l} = (\bar{l}_1, ..., \bar{l}_m)$ con $\bar{l}_i = l_i^* - \epsilon$ (lunghezze un po' più corte) tali per cui

$$\sum_{i=1}^{m} D^{-\bar{l}_i} = \sum_{i=1}^{m} D^{-l_i^*} D^{\epsilon} = 1$$

Ma

$$\sum_{i=1}^{m} p_i \bar{l}_i = \sum_{i=1}^{m} p_i (l_i^* - \epsilon) = \sum_{i=1}^{m} p_i l_i^* - \epsilon < \sum_{i=1}^{m} p_i l_i^*$$

e dunque l_i^* non sarebbe il minimo.
Per risolvere il problema, si usa il metodo dei moltiplicatori di Lagrange:

$$\begin{cases} f(l) = \sum_{i=1}^{m} p_i l_i = p \cdot l \\ \sum_{i=1}^{m} D^{-l_i} \leq 1 \end{cases} \quad \text{per } l = (l_1, ..., l_m) \in (\mathbb{R}^+)^m$$

Dunque

$$\frac{\partial}{\partial l_k} \left(\sum_{i=1}^{m} p_i l_i \right) = \lambda \frac{\partial}{\partial l_k} \left(\sum_{i=1}^{m} D^{-l_i} - 1 \right) \iff$$

$$\iff p_k = \lambda(-D^{-l_i} \ln D) \iff D^{-l_k} = -\frac{p_k}{\lambda \ln D}$$

Si inserisce nel vincolo

$$1 = \sum_{i=1}^{m} D^{-l_i} = -\sum_{i=1}^{m} \frac{p_i}{\lambda \ln D} = -\frac{\sum_{i=1}^{m} p_i}{\lambda \ln D} = -\frac{1}{\lambda \ln D} \Rightarrow \lambda = \frac{1}{\ln D}$$

Sostituendo si ottiene che $p_i = D^{-l_i{}^*}$
Si verifica che l_i^* è punto di minimo: $l^* = -\log_D p_i$
Poichè $f(l) \to \infty$ per $|l| \to \infty$, il suo minimo sarà ottenuto in un punto di minimo di $f_{|K}$, con K insieme compatto. Attraverso il procedimento è stato trovato un punto estremale (vincolato) l^*, dunque deve essere il punto di minimo vincolato.
Tutto ciò non risolve di fatto il problema, ma permette di dare una stima:

$$\min_{\text{C codice istantaneo per X}} L(C) = \min_{l \in (\mathbb{Z}^+)^m,\ \sum D^{-l_i} \leq 1} f(l) \geq \min_{l \in (\mathbb{R}^+)^m,\ \sum D^{-l_i} \leq 1} f(l) = f(l^*)$$

$$\sum_{i=1}^{m} p_i l_i^* = -\sum_{x \in \mathcal{X}} p(x) \log_D p(x) = H_D(X).$$

Da questa dimostrazione si ha

Teorema 2.11. *Per ogni codice istantaneo D-ario per la variabile aleatoria X con distribuzione p(x), vale*

$$L(C) \geq H_D(X).$$

Inoltre

$$L(C) = H_D(X) \iff l^*(x) = -\log_D p(x) \iff$$

$$\iff p(x) = D^{-l(x)} \text{ con } l(x) \in \mathbb{Z} \iff l_i^* = -\log_D p_i \in \mathbb{Z}^+$$

Se si ha proprio quest'ultimo caso, il codice si dice *super-ottimale*, e le lunghezze si indicano con l^{**}.

Definizione 2.12. Si definisce *codice di Shannon* il codice $C^{(S)}$ le cui lunghezze di parole di codice sono date da

$$l^{(s)}(x) = \lceil -\log_D p(x) \rceil$$

l'intero superiore, e le parole di codice sono date dalla famiglia $\{l_1^{(s)}, ..., l_m^{(s)}\}$ definite come sopra a partire dalle p_i

Definizione 2.13. Nel caso in cui $l_i^* \in \mathbb{Z}^+ \ \forall i$, il codice viene detto *super-ottimale* e le lunghezze vengono denotate con $l^{**} \in \mathbb{Z}^+$

2.3 Stima della bontà di un codice

Proposizione 2.14. *Dato un codice istantaneo D-ario con lunghezze $l_1, ..., l_m \in \mathbb{Z}^+$ e chiamati*

$$p = (p_1, ..., p_m) \quad e \quad r = \frac{1}{\sum_i D^{-l_i}}(D^{-l_1}, ..., D^{-l_m})$$

i vettori stocastici, si ha che

$$L(c) - H_D(X) = D_D(p||r) - \log_D(\sum_i D^{-l_i})$$

Dimostrazione.

$$L(c) - H_D(X) = \sum_i p_i l_i + \sum_i p_i \log_D p_i =$$

$$= -\sum_i p_i \log_D D^{-l_i} + \sum p_i \log_D p_i = \sum_i p_i \log_D \frac{p_i}{\left(\sum_j D^{-l_j}\right) r_i} =$$

$$= \sum_i p_i \log_D \frac{p_i}{r_i} - \sum_i p_i \log_D \left(\sum_j D^{-l_j}\right) = D_D(p||r) - \log_D \left(\sum_j D^{-l_j}\right)$$

□

Osservazione. Affinchè $L(c)$ raggiunga il limite teorico $H_D(X)$ è necessario che

$$\sum_i D^{-l_i} = 1 \Rightarrow \log_D \left(\sum_j D^{-l_j}\right) = \log_D(1) = 0$$

e che

$$p_i = \frac{D^{-l_i}}{\sum_j D^{-l_j}} = D^{-l_i} \Rightarrow p = r \Rightarrow D_D(p||r) = 0$$

Proposizione 2.15. *Si può trovare un codice istantaneo c tale che*

$$H_D(X) \leq L(C) < H_D(X) + 1$$

Dimostrazione. Il codice in questione è il codice di Shannon, dato da $l_i^{(s)} = \lceil -\log_D p_i \rceil$. Per le osservazioni precedenti, il codice di Shannon è istantaneo; inoltre per definizione di intero superiore

$$-\log_D p_i \leq l_i^{(s)} < -\log_D p_i + 1$$

Moltiplicando tutti i membri per p_i e poi sommando sugli indici i

$$-\sum_i p_i \log_D p_i \leq \sum_i p_i l_i^{(s)} < -\sum_i p_i \log_D p_i + \sum_i p_i \Rightarrow$$
$$\Rightarrow H_D(X) \leq L(C) < H_D(X) + 1$$

□

Osservazione. In realtà, se $H_D(X)$ è molto piccola, allora un solo bit è moltissimo e la stima non dice nulla sulla bontà del codice.
D'altra parte si può evitare questo problema ed essere più efficienti, comprimendo n-stringhe di cifre invece che una cifra alla volta; al posto della variabile X che prende valori in $\mathcal{X}$, si considera $X^n = (X_1, ..., X_n)$, a valori in $\mathcal{X}^n$, con distribuzione $p(x^n) = p(x_1) \cdot p(x_2) \cdot ... \cdot p(x_n)$.
Si costruisce un codice di Shannon per X^n, chiamato $C_n^{(s)}$. Allora

$$H_D(X^n) \leq L(C_n^s) < H_D(X^n) + 1$$

$$nH_D(X) \leq L(C_n^s) < nH_D(X) + 1$$

$$H_D(X) \leq \frac{L(C_n^s)}{n} < H_D(X) + \frac{1}{n}$$

dove il termine al centro è il consumo medio di bit per cifra.

Si pone ora una domanda: è possibile ottimizzare meglio la lunghezza di codice per una determinata variabile aleatoria X, usando codici univocamente identificabili anzichè i codici istantanei? Ovvero, è possibile che valga

$$\min_{C \ u.d.} L(C) < \min_{C \ ist.} L(C).$$

La risposta è negativa, per via del seguente lemma.

Lemma 2.16. *(Mc Millan) Le lunghezze di un codice univocamente identificabile soddisfano la disuguaglianza di Kraft.*

Dimostrazione. Dato il codice univocamente identificabile $C : \mathcal{X} \to \mathcal{D}^*$, si definisce la sua k-estensione $C^k : \mathcal{X}^k \to \mathcal{D}^k$:

$$C^K(x_1, x_2, ..., x_k) = C(x_1)C(x_2)...C(x_k)$$

Siccome C^*, l'estensione, è iniettivo per ipotesi, lo è anche C^K, essendo una sua restrizione.
Si nota ora che, dato l'insieme

$$a(n) = |\{x^k = (x_1, ..., x_k) : l^k(x^k) = n\}|$$

allora $a(n) \leq D^n \ \forall \ n$. Si può vedere inoltre che

$$l^k(x^k) = l(C^k(x^k)) = l(C(x_1)) + ... + l(C(x_k)) = l(x_1) + ... + l(x_k)$$

Dunque

$$\left(\sum_{x \in \mathcal{X}} D^{-l(x)} \right)^k = \sum_{x_1 \in \mathcal{X}} ... \sum_{x_k \in \mathcal{X}} D^{-l(x_1)} ... D^{-l(x_k)} =$$

$$= \sum_{x^k \in \mathcal{X}^k} D^{-l^k(x^k)} = \sum_{n=1}^{kl_{max}} a(n) D^{-n} \leq \sum_{n=1}^{kl_{max}} D^n D^{-n} = kl_{max}$$

dove l_{max} è la massima lunghezza di parola di codice per C, e kl_{max} è la massima lunghezza di parola di codice per C^k. Quindi

$$\sum_{x \in \mathcal{X}} D^{-l(x)} \leq (kl_{max})^{\frac{1}{k}}, \text{ con } \lim_{k \to \infty} = 1 \Longrightarrow \sum_{x \in \mathcal{X}} D^{-l(x)} \leq 1$$

□

2.4 Algoritmi di compressione

2.4.1 Codici di Huffmann

Si descrive ora un metodo per costruire un codice istantaneo, dovuto a David Albert Huffmann; tale metodo ottiene il minimo di $L(C)$.
Un primo fatto che va notato è che quando $m \leq D$, con $m = |\mathcal{X}|$ e D la cardinalità dell'alfabeto, un codice ottimale istantaneo è dato da $C(i\text{-esimo elemento di } \mathcal{X}) = i - 1$

Osservazione. Senza perdita di generalità, si può sempre assumere che

$$m = 1 + k(D - 1) \qquad k \in \mathbb{N}$$

Se così non fosse, si potrebbero aggiungere simboli fittizi con probabilità zero. La formula è interessante perchè la cardinalità di un albero D-ario in cui un nodo non ha figli o li ha tutti è in quella forma.

Definizione 2.17. Si definisce il codice di Huffman ricorsivamente su k in questo modo

- $k = 1$ Allora $C^{(H)}(i) = i - 1$
- $k - 1 \Rightarrow k \ (\Longleftrightarrow m = 1 + k(D - 1))$

Si riordinano gli m simboli in modo che le rispettive p_i siano ordinate per probabilità decrescente:

$$\sigma = (\sigma_1, ..., \sigma_m) \text{ tc } p_{\sigma_1} \geq p_{\sigma_2} \geq ... \geq p_{\sigma_m}$$

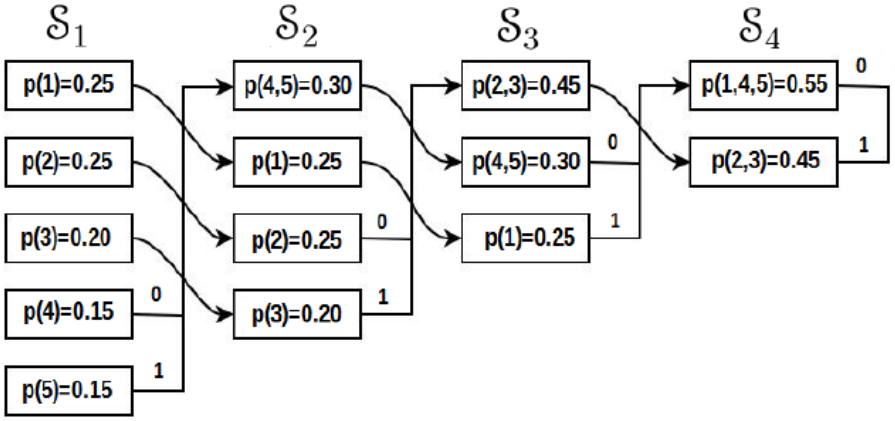
Si definisca un nuovo insieme $\mathcal{X}' = \{\sigma_1, \sigma_2, ..., \sigma_{m-D}, (m-D+1)'\}$, in cui l'ultimo è un nuovo simbolo, risultato dell'accoppiamento delle D cifre meno probabili. Ciò che si nota è che $|\mathcal{X}'| = m - (D-1) = 1 + (k-1)(D-1)$
A questi nuovi simboli si assegna il vettore stocastico di probabilità

$$p' = (p'_{\sigma_1}, ..., p'_{\sigma_m}, p'_{(m-D+1)'}) = (p_{\sigma_1}, ..., p_{\sigma_m}, \sum_{i=m-D+1}^{m} p_{\sigma_i})$$

Per procedura ricorsiva, si definisce il codice di Huffman $C^{(H)'}$ per la variabile aleatoria X' su $\mathcal{X}'$ con distribuzione p'. $C^{(H)}$ è definito come

$$C^{(H)}(\sigma_1) \begin{cases} C^{(H)'}(\sigma_1) & \text{per} i = 1, ..., m-D \\ C^{(H)'}((m-D+1)') + \text{"}(i-m+D-1)\text{"} & \text{per} i = m-D+1, .., m \end{cases}$$

Esempio 2.18. Sia $\mathcal{X} = \{1, 2, 3, 4, 5\}$ con probabilità rispettivamente 0.25, 0.25, 0.20, 0.15, 0.15, e sia $D = 2$. Si effettua il procedimento descritto combinando i due simboli meno probabili in un unico simbolo fino a rimanere con due soli simboli



Ripercorrendo la successione delle sorgenti ottenuta all'indietro, si ottiene il codice mostrato nella tabella sottostante.

x	$p(x)$	$l(x)$	codice
1	0,25	2	01
2	0,25	2	10
3	0,2	2	11
4	0,15	3	000
5	0,15	3	001

Definizione 2.19. Una definizione analoga di codice di Huffman è

$$c^{(H)}(k) := \begin{cases} c^{(H)'}(k) & \text{per } k = \sigma_i,\ 1 \leq i \leq m - D \\ c^{(H)'}((m - D + 1)') + "j" & \text{per } k = \sigma_i,\ m - D + 1 \leq i \leq m, \\ & \text{con } j = i - m + D - 1 \end{cases}$$

Lemma 2.20. *Sia data la variabile aleatoria X con distribuzione $p = (p_1, ..., p_m)$. Sia $|\mathcal{X} = 1 + k(D-1), k \in \mathbb{Z}^+$. Allora esiste un codice istantaneo ottimale tale che:*

1. *$p_j > p_i \Rightarrow l_j < l_i$*
2. *nell'albero finito che descrive il codice, ogni diramazione ha tutti i suoi figli oppure nessuno*

Dimostrazione. 1.) Esiste un codice ottimale per X, basta restringersi per esempio a

$$l = (l_1, ..., l_m) \in \{1, 2, ..., m\}^m$$

Sicuramente in quest'insieme è presente ogni l_i, dunque posso trovare il minimo per l. Per la 2.9, esiste un codice che ha come lunghezze il l'insieme di minimo di lunghezze trovato sopra. Dunque esiste un codice istantaneo ottimale su $\mathcal{X}$ con $|\mathcal{X}| = 1 + k(D-1)$
Se per assurdo non valesse la proprietà 1, esisterebbero indici i, j con $p_i > p_j$ e $l(\sigma_j) < l(\sigma_i)$. Si definisce un nuovo insieme di lunghezze $l' = (l'_1, ..., l'_m)$

$$l'_k = \begin{cases} l_j & \text{se } k = i \\ l_i & \text{se } k = j \\ l_k & \text{altrimenti} \end{cases}$$

invertendo così le lunghezze. Si ottiene un nuovo codice con insieme di lunghezze l' tale che

$$\sum_k p_k l_k - \sum_k p_k l'_k = p_i l_i + p_j l_j - p_i l'_i - p_j l'_j = p_i l_i + p_j l_j - p_i l_j - p_j l_i =$$

$$= p_j(l_j - l_i) + p_i(l_i - l_j) = p_j(l_j - l_i) - p_i(l_j - l_i) = (p_j - p_i)(l_j - l_i) > 0$$

Il nuovo codice con lunghezza l' sarebbe strettamente migliore di quello con l, quindi il codice l di partenza non era ottimale: assurdo.
2.) Si supponga che non siano state introdotte cifre fittizie, con probabilità nulla, e che la seconda asserzione non sia vera: dunque esiste almeno una diramazione che ha dei figli, ma non tutti i possibili D figli. Si prenda una di queste diramazioni; non può avere un solo ramo discendente, altrimenti si potrebbe eliminare tale parte della diramazione, andando ad "alzare" tutti i discendenti, in sostanza andando ad accorciare di un D-it la lunghezza di tutti i simboli collegati a tale discendenza, tenendo comunque costante la lunghezza di restanti simboli. Tale procedimento andrebbe a creare un codice con lunghezza media minore, andando contro l'ipotesi secondo la quale il codice da cui si partiva era già ottimale.
Dunque, il nodo considerato ha un numero di figli compreso tra 2 e $D - 1$. Ricordando che per l'albero vale la relazione $m = 1 + k(D - 1)$, tale nodo non può essere l'unico non completo di tutti i possibili figli, perchè se lo fosse, si potrebbe pensare di aggiungere i j figli mancanti, con j compreso tra 1 e $D - 2$, con determinate probabilità, ottenendo così un albero con tutti i nodi completi; questo avrà cardinalità

$$m + j = 1 + k(D - 1) + j$$

e per rimanere un albero che rappresenta un codice di Huffmann dovrebbe essere della forma $1 + k'(D - 1)$, evidentemente impossibile.
Rimangono dunque due casi:

a I nodi non completi sono su più livelli, ovvero i nodi non completi si trovano a distanze diverse dalla radice;

b Tutti i nodi non completi sono sullo stesso livello, ovverosia si trovano alla stessa distanza dalla radice.

Nel primo caso, sarebbe comunque sempre possibile prendere un discendente e spostarlo dove non vi è ancora un nodo completo a distanza dalla radice inferiore, andando così ad accorciare la lunghezza media di codice.
Nel secondo caso posso spostare i discendenti sui nodi dello stesso livello, fino a rendere completi tutti i nodi dell'albero da sinistra verso destra, lasciando al più un nodo non completo. Tale operazione non modifica la lunghezza media di codice. Se così facendo ci si ritrova nel caso di un nodo con meno di D figli, ci si ricollega al

caso già trattato sopra, a cui si arriva ad un assurdo. Dunque, l'unica possibilità rimasta, è quella descritta dalla condizione 2.
Nel caso ci siano delle cifre fittizie per soddisfare il vincolo su m, i procedimenti di modifica dell'albero adottati nella precedente trattazione non aumentano la lunghezza media di codice; quindi, dato un codice ottimale che non soddisfa la condizione 2, si possono aggiungere cifre con probabilità nulla per costruire un nuovo codice, ottimale, che la rispetti. □

Proposizione 2.21. *Il codice di Huffmann è ottimale*

Dimostrazione. Si dimostra per induzione su k, ricordando che $m = 1 + k(D-1)$
Per $k = 1$ si ottiene che $m = D$, e dunque è banalmente il codice ottimale
Per il passo induttivo, si prenda un codice istantaneo C e si voglia dimostrare che

$$L(C) \geq L(C^{(H)})$$

Prendendo il codice canonico usato C_C usato nel precedente lemma, vale che $L(C) \geq L(C_C)$. Basterà dimostrare che $L(C_C) \geq L(C^{(H)})$.
Si osserva che sia $C^{(H)}$ che C_C hanno la proprietà che le D parole di codice meno probabili differiscono solo per l'ultimo D-it e dunque corrispondono a rami terminali dello stesso nodo. Tale proprietà di $C^{(H)}$ è data per costruzione, mentre per C_C discende dalle due proprietà del lemma sopracitato.
Si consideri ora l'alfabeto di cifre $\mathcal{X}'$ come nella definizione del codice di Huffmann, con la corrispondente distribuzione di probabilità p'. Da qui si costruisce un codice C'_C:

$$C'_C(k) = \begin{cases} C_C(k) & \text{se } k = \sigma_i \text{ con } 1 \leq i \leq m-D \\ C_C(\sigma_i) - \text{"ultimo } D\text{-it} & \text{se } k = (m-D+1)' \text{ con } m-D+1 \leq i \leq m \end{cases}$$

Ricordando come è fatta la distribuzione p' su $\mathcal{X}$

x'	$p'(x')$
σ_1	p_{σ_1}
σ_2	p_{σ_2}
$\vdots$	$\vdots$
σ_{m-D}	$p_{\sigma_{m-D}}$
$(m-D+1)'$	$p_{(m-D+1)'} = \sum_{i=m-D+1}^{D} p_{\sigma_i}$

Si può trovare una formula che leghi $L(C'_C)$ a $(L(C_C)$.

$$L(C'_C) = \sum_{i=1}^{m-D} p_{\sigma_i} l(C_C(\sigma_i)) + p_{(m-D+1)'} l(C'_C((m-D+1)')) =$$

$$= \sum_{i=1}^{m-D} p_{\sigma_i} l(C_C(\sigma_i)) + \left(\sum_{i=m-D+1}^{D} p_{\sigma_i} \right) (l(C_C(\sigma_i)) - 1) =$$

$$= \sum_{i=1}^{m-D} p_{\sigma_i} l(C_C(\sigma_i)) + \sum_{i=m-D+1}^{D} p_{\sigma_i} (l(C_C(\sigma_i)) - \sum_{i=m-D+1}^{D} p_{\sigma_i} =$$

$$= \sum_{i=1}^{m} p_{\sigma_i} l(C_C(\sigma_i)) - p_{(m-D+1)'} = L(C_C) - p_{(m-D+1)'}$$

Un conto analogo si può fare per $L(C^{(H)'})$ rispetto a $L(C^{(H)})$:

$$L(C^{(H)'}) = L(C^{(H)}) - p_{(m-D+1)'}$$

Per ipotesi induttiva si ha

$$L(C^{(H)'}) \leq L(C'_C)$$

e dunque

$$L(C^{(H)'}) \leq L(C'_C) \iff L(C^{(H)'}) + p_{(m-D+1)'} \leq L(C'_C) + p_{(m-D+1)'} \iff$$

$$\iff L(C^{(H)}) \leq L(C_C)$$

□

2.4.2 Trasformata di Burrows-Wheeler

La trasformata di Burrows-Wheeler, abbreviata a BWT, è un algoritmo utilizzato in moltissime applicazioni per la compressione dati. La trasformata che costituisce gran parte dell'algoritmo, fu sviluppata da Wheeler nel 1983 e viene attualmente utilizzata come primo stadio di trasformazione della stringa da comprimere in molti programmi di compressione commerciali.
Con la trasformata BWT è necessario avere a disposizione l'intera sequenza codificata prima di iniziare la fase di decodifica. Il metodo di compressione è una combinazione dei due seguenti algoritmi:

- *trasformata di Burrows-Wheeler*: se applicata ad una stringa di caratteri non effettua una vera e propria compressione, ma permuta in maniera reversibile l'ordine du questi. Se la stringa originale contiene molte sottostringhe che si ripetono speso, la stringa trasformata ha numerose parti in cui un carattere viene ripetuto diverse volte di fila

- *Move-To-Front*: è un semplice metodo di compressione utile a comprimere stringhe caratterizzate da simboli ripetuti

Data una sequenza $x_1x_2...x_N$ di lunghezza N, se ne creano altre $N-1$, ciascuna delle quali è una permutazione ciclica della sequenza originale. Si ottiene la i-esima rotazione concatenando la sottostringa dei primi i caratteri alla sottostringa degli ultimi $N-i$, ottenendo le seguenti permutazioni della stringa di partenza

$$x_{i+1}x_{i+2}...x_Nx_1x_2...x_i \qquad \forall\ i = 1, 2, ..., N-1$$

Si riordinano le N sequenze in ordine lessicografico. Il codificatore invia la sequenza di lunghezza N che si compone prendendo l'ultimo simbolo di ogni rotazione. Sia L l'ultima colonna di M e I l'indice di riga che denota la stringa originale. Questi sono i dati che bisogna inviare al destinatario. Questi svolgerà l'algoritmo di anti-trasformazione.
Come primo passo, egli calcola la prima colonna della matrice M, ordinando i caratteri del vettore L, ottenendo così il vettore F. Si immagini di definire la matrice M' come la matrice ottenuta ruotando di una posizione vero destra le righe di M.

$$M'[i, j] = M[i, (j-1) \mod N]$$

Chiaramente questa matrice è sconosciuta a chi riceve il messaggio, ma facilita la spiegazione del funzionamento dell'algoritmo.
In M' le righe sono ordinate a partire dal secondo carattere; si può quindi dedurre che ogni sottoinsieme di righe di M' che iniziano con lo stesso carattere appare in M' in ordine lessicografico. Ovviamente la stessa proprietà valeva anche per la matrice M.
Si chiami T il vettore che indica la corrispondenza tra le righe delle matrici M e M': la n-esima riga di M' corrisponde alla $T[n]$-esima riga di M; tale vettore verrà calcolato in seguito.
Si nota che se $L[j]$ è la k-esima istanza di un carattere in L, allora, definito $T[j] = i$, $F[i]$ è la k-esima istanza dello stesso carattere in F. Sostituendo, si ottiene $F[T[j]] =$

$L[j]$.
Ora, per ogni $i = 0, ..., N-1$, i caratteri $L[i]$ e $F[i]$ sono rispettivamente l'ultimo e il primo carattere della riga i di M: quindi il carattere $L[i]$ precede ciclicamente il carattere $F[i]$ in S.

$$S = "...L[i]F[i]..."$$

Sostituendo $i = T[j]$ nella relazione ricavata in precedenza, si ottiene che $L[T[j]]$ precede ciclicamente $L[j]$ in S.

$$S = "...L[T[j]]F[T[j]]..." = "...L[T[j]]L[j]..." =$$

$$= "...F[T[T[j]]]L[j]...""...F[T^2[j]]L[j]..."$$

Possiamo ripetere il ragionamento fatto sopra, $L[T^2[j]]$ precede ciclicamente $F[T^2[j]]$.

$$S = "...L[T^2[j]]F[T^2[j]]L[j]..." = "...L[T^2[j]]L[T[j]]L[j]..."$$

Iterando il procedimento, si ottiene

$$S = "L[T^{N-1}[j]]...L[T[j]]L[j]"$$

ottenendo così la regola generale

$$S[N-1-i] = L[T^i[I]] \text{ (dove } T^0[x] = x)$$

dal quale si evince che basta sapere l'ultima lettera della parola per ricostruirla interamente con l'ausilio del vettore T.
Per mezzo di F e L si calcola il vettore T, che si ricorda essere il vettore che descrive le permutazioni delle righe della matrice M' nella matrice M. Si unisce il vettore L al vettore F e si nota che, ovviamente, sono ordinati rispetto alla colonna F. Si assegnano ad ogni riga indici da 0 a $N-1$, secondo l'ordine lessicografico rispetto alla prima colonna, quindi L. Se ci sono due righe identiche, gli indici si assegnano in ordine dall'alto verso il basso, poichè tali righe erano in ordine alfabetico prima secondo la colonna F, perciò lo sono ancora ora.
Ora ci sono tutti gli elementi che occorrono per ricostruire il messaggio originale. Si parte da $L[2]$, l'ultimo carattere di S. Con il vettore T si ricavano tutti i caratteri predecessori di $L[I]$. Concluse le iterazioni, si ottiene la stringa originaria S.

Esempio 2.22. Si utilizzi la stringa campione S = "ABRACADABRA" e si costruisca una matrice N x N, chiamata J, le cui righe sono tutte le possibili rotazioni di S, di caratteri $S[0], ..., S[N-1]$

$$
W = \begin{matrix}
A & B & R & A & C & A & D & A & B & R & A \\
B & R & A & C & A & D & A & B & R & A & A \\
R & A & C & A & D & A & B & R & A & A & B \\
A & C & A & D & A & B & R & A & A & B & R \\
C & A & D & A & B & R & A & A & B & R & A \\
A & D & A & B & R & A & A & B & R & A & C \\
D & A & B & R & A & A & B & R & A & C & A \\
A & B & R & A & A & B & R & A & C & A & D \\
B & R & A & A & B & R & A & C & A & D & A \\
R & A & A & B & R & A & C & A & D & A & B \\
A & A & B & R & A & C & A & D & A & B & R
\end{matrix}
$$

Si riordinino in ordine lessicografico le righe rispetto alla prima colonna e si indicizzino le righe, mettendo in evidenza l'ultima colonna e l'indice della riga originale

$$
\begin{matrix}
0 & & A & A & B & R & A & C & A & D & A & B & R \\
1 & & A & B & R & A & A & B & R & A & C & A & D \\
2 & & A & B & R & A & C & A & D & A & B & R & A \\
3 & & A & C & A & D & A & B & R & A & A & B & R \\
4 & & A & D & A & B & R & A & A & B & R & A & C \\
5 & M = & B & R & A & A & B & R & A & C & A & D & A \\
6 & & B & R & A & C & A & D & A & B & R & A & A \\
7 & & C & A & D & A & B & R & A & A & B & R & A \\
8 & & D & A & B & R & A & A & B & R & A & C & A \\
9 & & R & A & A & B & R & A & C & A & D & A & B \\
10 & & R & A & C & A & D & A & B & R & A & A & B
\end{matrix}
$$

Sia $L =$ "RDARCAAAABB" e $I = 2$ e dunque $F =$"AAAAABBCDRR". Sotto si può osservare la matrice M'.

$$
\text{M'} = \begin{matrix}
\text{R} & \text{A} & \text{A} & \text{B} & \text{R} & \text{A} & \text{C} & \text{A} & \text{D} & \text{A} & \text{B} \\
\text{D} & \text{A} & \text{B} & \text{R} & \text{A} & \text{A} & \text{B} & \text{R} & \text{A} & \text{C} & \text{A} \\
\text{A} & \text{A} & \text{B} & \text{R} & \text{A} & \text{C} & \text{A} & \text{D} & \text{A} & \text{B} & \text{R} \\
\text{R} & \text{A} & \text{C} & \text{A} & \text{D} & \text{A} & \text{B} & \text{R} & \text{A} & \text{A} & \text{B} \\
\text{C} & \text{A} & \text{D} & \text{A} & \text{B} & \text{R} & \text{A} & \text{A} & \text{B} & \text{R} & \text{A} \\
\text{A} & \text{B} & \text{R} & \text{A} & \text{A} & \text{B} & \text{R} & \text{A} & \text{C} & \text{A} & \text{D} \\
\text{A} & \text{B} & \text{R} & \text{A} & \text{C} & \text{A} & \text{D} & \text{A} & \text{B} & \text{R} & \text{A} \\
\text{A} & \text{C} & \text{A} & \text{D} & \text{A} & \text{B} & \text{R} & \text{A} & \text{A} & \text{B} & \text{R} \\
\text{A} & \text{D} & \text{A} & \text{B} & \text{R} & \text{A} & \text{A} & \text{B} & \text{R} & \text{A} & \text{C} \\
\text{B} & \text{R} & \text{A} & \text{A} & \text{B} & \text{R} & \text{A} & \text{C} & \text{A} & \text{D} & \text{A} \\
\text{B} & \text{R} & \text{A} & \text{C} & \text{A} & \text{D} & \text{A} & \text{B} & \text{R} & \text{A} & \text{A}
\end{matrix}
$$

In questo caso, l'ultima lettera è identificata da $L[2]$, poichè 2 è l'indice che rappresenta la riga della matrice M dove si trova la parola originale non permutata.
Si costruisce il vettore T. Il risultato è il seguente

9	R	A
8	D	A
0	A	A
10	R	A
7	C	A
1	A	B
2	A	B
3	A	C
4	A	D
5	B	R
6	B	R

Il vettore è $T = [9, 8, 0, 10, 7, 1, 2, 3, 4, 5, 6]$

Per mezzo del vettore T si ricava la stringa originaria S:

$$
\begin{aligned}
&S[10] = L[2] = \text{'a'} \\
&S[9] = L[T[2]] = L[0] = \text{'r'} \\
&S[8] = L[T^2[2]] = L[9] = \text{'b'} \\
&S[7] = L[T^3[2]] = L[5] = \text{'a'} \\
&S[6] = L[T^4[2]] = L[1] = \text{'d'} \\
&S[5] = L[T^5[2]] = L[8] = \text{'a'} \\
&S[4] = L[T^6[2]] = L[4] = \text{'c'} \\
&S[3] = L[T^7[2]] = L[7] = \text{'a'} \\
&S[2] = L[T^8[2]] = L[3] = \text{'r'} \\
&S[1] = L[T^9[2]] = L[10] = \text{'b'} \\
&S[0] = L[T^{10}[2]] = L[6] = \text{'a'}
\end{aligned}
$$

Va notato che la sequenza $T^i[I]$ per $i = 0, ..., N-1$, non sempre è una permutazione dei numeri $0, ..., N-1$: ciò accade quando S è una ripetizione di una sottostringa della forma Z^p per $p > 1$; in tal caso $T^i[I]$ per $i = 0, ..., N-1$ diventa della forma Z'^p per qualche sottosequenza Z'.
Ad esempio, se S="FUGGIFUGGI", Z ="FUGGI" e $p = 2$. Si calcoli la sequenza $T^i[I]$

$$S = \text{"FUGGIFUGGI"} \rightarrow L = \text{"IIUUGGGGFF"} \quad \text{e} \quad I = 1$$

$$F = \text{"FFGGGGIIUU"} \text{ e } T = [6, 7, 8, 9, 2, 3, 4, 5, 0, 1]$$

$$
\begin{aligned}
S[9] &= L[1] = \text{'i'} \\
S[8] &= L[T[1]] = L[7] = \text{'g'} \\
S[7] &= L[T^2[1]] = L[5] = \text{'g'} \\
S[6] &= L[T^3[1]] = L[3] = \text{'u'} \\
S[5] &= L[T^4[1]] = L[9] = \text{'f'} \\
S[4] &= L[T^5[1]] = L[1] = \text{'i'} \\
S[3] &= L[T^6[1]] = L[7] = \text{'g'} \\
S[2] &= L[T^7[1]] = L[5] = \text{'g'} \\
S[1] &= L[T^8[1]] = L[3] = \text{'u'} \\
S[0] &= L[T^9[1]] = L[9] = \text{'f'}
\end{aligned}
$$

Esempio 2.23. Si utilizzi sempre una stringa campione: $S =$"CAPANNA". Si aggiunge un carattere finale assumendo che questo non compaia all'interno del testo, al fine di distinguere l'ultimo carattere quando si ricostruisce la stringa. Si ottiene così la nuova stringa $S =$"CAPANNA@". Si costruisce una tabella con tutte le rotazioni della stringa, come nell'esempio precedente, e poi si ordina in ordine alfabetico rispetto alla prima colonna

$$
\begin{array}{cccccccc}
C & A & P & A & N & N & A & @ \\
A & P & A & N & N & A & @ & C \\
P & A & N & N & A & @ & C & A \\
A & N & N & A & @ & C & A & P \\
N & N & A & @ & C & A & P & A \\
N & A & @ & C & A & P & A & N \\
A & @ & C & A & P & A & N & N \\
@ & C & A & P & A & N & N & A
\end{array}
\Rightarrow
\left|\begin{array}{cccccccc}
A & N & N & A & @ & C & A & P \\
A & P & A & N & N & A & @ & C \\
A & @ & C & A & P & A & N & N \\
C & A & P & A & N & N & A & @ \\
N & A & @ & C & A & P & A & N \\
N & N & A & @ & C & A & P & A \\
P & A & N & N & A & @ & C & A \\
@ & C & A & P & A & N & N & A
\end{array}\right|
$$

Come prima, la stringa che si invierà al destinatario è l'ultima colonna. Questo è tutto ciò che gli serve per svolgere l'algoritmo di antitrasformazione.
Il destinatario costruisce una tabella, inizialmente vuota, di righe e colonne pari al numero di caratteri della stringa con aggiunto il simbolo ausiliario. Egli inserisce come ultima colonna della matrice vuota la stringa inviatagli dal mittente. Dopodichè, la riordina, dall'alto al basso. Successivamente gli affianca, come penultima colonna, la stessa stringa ricevuta. Ora riordina le due colonne secondo l'ordine

alfabetico rispetto alla penultima colonna, e così via, finchè non avrà riempito tutta la tabella.

P		A		P	A		A	N			P	A	N	N	A	@	C	A
C		A		C	A		A	P			C	A	P	A	N	N	A	@
N		A		N	A		A	@			N	A	@	C	A	P	A	N
@	⇒	C	⇒	@	C	⇒	C	A	⇒ ... ⇒		@	C	A	P	A	N	N	A
N		N		N	N		N	A			N	N	A	@	C	A	P	A
A		N		A	N		N	N			A	N	N	A	@	C	A	P
A		P		A	P		P	A			A	P	A	N	N	A	@	C
A		@		A	@		@	C			A	@	C	A	P	A	N	N

La riga ottenuta che comincia per il carattere ausiliario contiene la stringa originale, scritta con i caratteri in ordine corretto.

Una volta codificato il messaggio, bisogna mandarlo in maniera efficace, attraverso l'algoritmo MTF; esso non comprime i dati ma aiuta a ridurre la ridondanza al loro interno, specie dopo l'applicazione della trasformazione BWT, dopo la quale un simbolo che è comparso da poco compare di nuovo con alta probabilità. Il metodo MTF funziona nel seguente modo:

- si inizializza un vettore di lunghezza pari alla cardinalità dell'alfabeto contenente i simboli utilizzati. Questo vettore rappresenta in ogni momento una permutazione di tutti gli elementi dell'alfabeto.
- si leggono in serie i caratteri dell'input. Per ognuno viene dato in output la sua corrente posizione all'interno del vettore costruito, in seguito si modifica il vettore: il simbolo corrente viene rimosso dalla posizione e reinserito in testa al vettore.
- si codifica con il metodo di Huffmann, codifica aritmetica o RLE la stringa ottenuta in output e spedita al destinatario, che decodificherà il messaggio e applicherà l'antitrasformazione BWT

Esempio 2.24. Riprendiamo il primo esempio, dove $S =$"PCN@NAAA". Si prenda in considerazione, per semplicità, l'alfabeto italiano a 21 lettere più una per il carattere ausiliario.
Il vettore iniziale sarà

$$A, B, C, D, E, F, G, H, I, L, M, N, O, P, Q, R, S, T, U, V, Z, @$$

La prima lettera della sequenza da codificare è **P**, che compare all'indice 13 del vettore numerato da 0 a 21. Si scrive "13" nella stringa di output. Il carattere **P** si sposta all'inizio della lista e si procede con il nuovo vettore

$$P, A, B, C, D, E, F, G, H, I, L, M, N, O, Q, R, S, T, U, V, Z, @$$

Il carattere successivo è **C**, che ora compare all'indice 3. Si aggiunge 3 alla sequenza di output, si sposta **C** in cima alla lista

$$C, P, A, B, D, E, F, G, H, I, L, M, N, O, Q, R, S, T, U, V, Z, @$$

e così via.
Continuando in questo modo, si ottiene il codice dell'intera sequenza: 13, 3, 12, 21, 1, 4, 0, 0. A quest'ultimo si va ad applicare il metodo di compressione che più si preferisce. Il procedimento da eseguire è esplicitato in tabella

Iterazione	Lista alfabeto	Codice sequenza
PCN@NAAA	ABCDEFGHILMNOPQRSTUVZ@	13
P**C**N@NAAA	PABCDEFGHILMNOQRSTUVZ@	13, 3
PC**N**@NAAA	CPABDEFGHILMNOQRSTUVZ@	13, 3, 12
PCN**@**NAAA	NCPABDEFGHILMOQRSTUVZ@	13, 3, 12, 21
PCN@**N**AAA	@NCPABDEFGHILMOQRSTUVZ	13, 3, 12, 21, 1
PCN@N**A**AA	N@CPABDEFGHILMOQRSTUVZ	13, 3, 12, 21, 1, 4
PCN@NA**A**A	AN@CPBDEFGHILMOQRSTUVZ	13, 3, 12, 21, 1, 4, 0
PCN@NAA**A**	AN@CPBDEFGHILMOQRSTUVZ	13, 3, 12, 21, 1, 4, 0, 0

Il destinatario del messaggio, una volta decodificato il messaggio con il metodo usato dal mittente, per convertire il vettore numerico in lettere dell'alfabeto, dovrà adottare il MTF al contrario
Ad ogni numero del vettore, associa la corrispondente lettera dell'alfabeto. Segnata la prima lettera, aggiorna il vettore alfabeto, ponendo all'inizio la prima lettera decodificata. Leggerà poi la seconda, la segnerà, la porrà in prima posizione del vettore alfabeto, e così via

Codice sequenza	Lista alfabeto	Decodifica
13, 3, 12, 21, 1, 4, 0, 0	ABCDEFGHILMNOPQRSTUVZ@	P
3, 12, 21, 1, 4, 0, 0	PABCDEFGHILMNOQRSTUVZ@	P, C
12, 21, 1, 4, 0, 0	CPABDEFGHILMNOQRSTUVZ@	P, C, N
21, 1, 4, 0, 0	NCPABDEFGHILMOQRSTUVZ@	P, C, N, @
1, 4, 0, 0	@NCPABDEFGHILMOQRSTUVZ	P, C, N, @, N
4, 0, 0	N@CPABDEFGHILMOQRSTUVZ	P, C, N, @, N, A
0, 0	AN@CPBDEFGHILMOQRSTUVZ	P, C, N, @, N, A, A
0	AN@CPBDEFGHILMOQRSTUVZ	P, C, N, @, N, A, A, A

Analogamente si può procedere con tale metodo anche se si è usato il la prima modalità di codifica di BWT, è sufficiente aggiungere alla fine delle diverse parole da trasmettere il simbolo ausiliario @.

Il codificatore MTF rappresenta con numeri poco elevati caratteri che si sono presentati recentemente; considerato quello che ci si aspetta come output dalla trasformazione BWT, MTF avrà ottime probabilità di produrre una gran quantità di valori vicini allo 0, dando ottimo materiale al compressore che segue.
In pratica l'algoritmo BWT ordina alfabeticamente tutte le possibili rotazioni di una stringa di input S e genera una nuova stringa che consiste nell'ultimo carattere di ciascuna rotazione. Per capire il motivo per cui la stringa così ottenuta è di più facile compressione, si consideri l'effetto dato da una singola lettera appartenente ad una parola particolarmente comune in un blocco di una lingua, in inglese, per esempio. Si utilizzi l'esempio della lettera **t** all'interno della parola **the** e si assuma che una stringa di input contenga più volte la parola **the**. Quando la lista delle rotazioni della stringa in input viene ordinata, tutte le rotazioni che iniziano con **he** si troveranno in righe vicine e una buona parte di queste righe finirà con la lettera **t**. Una regione della nuova stringa conterrà quindi una forte concentrazione di caratteri uguali a **t**, mescolati con altri caratteri che usualmente precedono la stringa **he** in inglese, come **s**, **T** o **S**. Analogo discorso può essere applicato agli altri caratteri in tutte le parole, si avrà quindi che ogni regione localizzata della stringa permutata è tale da contenere un alto numero di pochi caratteri distinti. L'effetto generale è che la probabilità che un dato carattere compaia in un dato punto della stringa perturbata è molto alta se tale carattere compare in un punto vicino al punto dato, e bassa se questo non accade.

3 Codici a blocchi

Si tratta di codici univocamente decodificabili e risultano particolarmente semplici da implementare. Nei capitoli successivi si studieranno codici a blocchi particolare, dotati di ulteriori proprietà (linearità) che consentono di gestirli meglio.

Esempio 3.1. In Italia, come in molti altri Paesi, i cittadini dispongono di un Codice di Identificazione Fiscale. Tale codice identifica in modo univoco e sin dalla nascita le persone che sono iscritte ai registri dell'anagrafe tributaria. E' un codice alfanumerico di 16 caratteri; I primi 15 relativi ai dati personali: le prime tre consonanti del cognome, la prima, la terza e la quarta consonante del nome le ultime due cifre dell'anno di nascita, il mese di nascita codificato con una lettera dell'alfabeto, le due cifre del giorno di nascita (a cui va aggiunto 40 se si è di sesso femminile) e il codice del comune di nascita. L'ultimo è un carattere di controllo, che può anche servire a distinguere due codici fiscali identici nella parte dei dati personali. Tale carattere si calcola nel seguente modo: a ciascuno dei primi 15 simboli viene assegnato un numero secondo la tabella sottostante.

Caratteri posizione pari

0=0	1=1	2=2	3=3	4=4	5=5	6=6	7=7	8=8	9=9	A=0	B=1	C=2
D=3	E=4	F=5	G=6	H=7	I=8	J=9	K=10	L=11	M=12	N=13	O=14	P=15
Q=16	R=17	S=18	T=19	U=20	V=21	W=22	X=23	Y=24	Z=25			

Caratteri posizione dispari

0=1	1=0	2=5	3=7	4=9	5=13	6=15	7=17	8=19	9=21	A=1	B=0	C=5
D=7	E=9	F=13	G=15	H=17	I=19	J=21	K=2	L=4	M=18	N=20	O=11	P=3
Q=6	R=8	S=12	T=14	U=16	V=10	W=22	X=25	Y=24	Z=23			

Carattere di controllo

0=A	1=B	2=C	3=D	4=E	5=F	6=G	7=H	8=I	9=J	10=K	11=L	12=M
13=N	14=O	15=P	16=Q	17=R	18=S	19=T	20=U	21=V	22=W	23=X	24=Y	25=Z

I valori così determinati vengono addizionati tra loro e la somma divisa per 26. Il sedicesimo carattere rappresenta il resto della divisione della somma prima ottenuta per 26, e assegnando al numero ottenuto una lettera dell'alfabeto (A per resto 0, B per resto 1 e così via).
Si prenda il codice fiscale del candidato: GRGFNC96S15F861B. Si suddividano ora in posizioni pari e dispari. I rispettivi valori sono riportati nella tabella sottostante.

Dispari	Valore	Pari	Valore
G	15	R	17
G	15	F	5
N	20	C	2
9	21	6	6
S	12	1	1
5	13	F	5
8	19	6	6
1	0	B	1

Si ha

$$(15 + 17 + 15 + 5 + 20 + 2 + 21 + 6 + 12 + 1 + 13 + 5 + 19 + 6 + 0) \equiv 1 \mod 26$$

3.1 Blocchi di informazione

In un codice *a blocchi* il flusso di dati da trasmettere viene codificato in pacchetti di unità discrete, ognuna contenente n caratteri, i blocchi appunto. La proprietà fondamentale di questo tipo di codici è che ogni pacchetto in cui viene suddiviso il messaggio completo, è codificato e decodificato in modo indipendente e autonomo sia dai pacchetti che lo precedono che da quelli che lo seguono. Per descrivere l'azione generale dell'apparecchiatura di codifica e decodifica basta descrivere il comportamento della stessa su uno solo di essi.

Definizione 3.2. Un *codice a blocchi* $\mathcal{C}$ *di lunghezza* n *con* M *parole di codice su un alfabeto* $\mathcal{A}$ è un insieme contenente M elementi, ognuno di lunghezza n, in cui ogni carattere di ogni elemento appartiene all'insieme $\mathcal{A}$.

Per un siffatto codice si userà il termine (n, M)-codice sopra $\mathcal{A}$ o, più semplicemente, quantunque non ci sai ambiguità sulla scelta dell'alfabeto, solo (n, M)-codice

Esempio 3.3. Il codice ASCII è un (7,128)-codice a blocchi

Esempio 3.4. L'insieme

$$\mathcal{C} = \{(00000), (10110), (01011), (11101)\}$$

è un (5,4)-codice binario

3.2 Metrica di Hamming

Un elemento fondamentale per la costruzione di un codice correttore è la possibilità di quantificare il livello di somiglianza fra due parole di lunghezza n. Questo viene realizzato introducendo una distanza sull'insieme $V = \mathcal{A}^n$.

Definizione 3.5. La distanza $d(a, b)$ di $a = (a_1, ..., a_n)$ e $b = (b_1, ..., b_n)$ è il numero di indici i tale che a_i è diverso da b_i:

$$d(a, b) = |\{i : a_i \neq b_i\}|$$

Valgono le seguenti, per $a, b, c \in V$; le prime tre mostrano che d è una distanza in senso topologico:

1. $d(a, b) = 0 \iff a = b$;
2. $d(a, b) = d(b, a)$
3. $d(a, b) \leq d(a, c) + d(c, b)$;
4. $d(a, b) = d(a - b, 0)$.

Dimostrazione. 1. $d(a, b) = 0$ vuol dire che per nessun indice i si ha che $a_i \neq b_i$, dunque $a = b$
2. Banale
3. Sia $\Delta(x, y) = \{i : x_i \neq_i\}$, sicchè $d(x, y) = |\Delta(x, y)|$. Se $i \in \Delta(a, b)$, allora $a_i \neq b_i$ e dunque non può essere contemporaneamente $a_i = c_i$ e $c_i = b_i$. Dunque $i \in \Delta(a, c)$ oppure $i \in \Delta(c, b)$, ovvero $i \in \Delta(a, c) \cup \Delta(c, b)$. Dunque $\Delta(a, b) \subseteq \Delta(a, c) \cup \Delta(c, b)$, da cui

$$d(a, b) = |\Delta(a, b)| \leq |\Delta(a, c) \cup \Delta(c, b)| \leq |\Delta(a, c)| + |\Delta(c, b)| = d(a, c) + d(c, b)$$

4. $d(a, b) = |\{i : a_i \neq b_i\}| = |\{i : a_i - b_i \neq 0\}| = |\{i : (a - b)_i \neq 0\}| = d(a - b, 0)$. □

Definizione 3.6. Sia $\mathcal{C}$ un codice, formato da parole della medesima lunghezza n su un alfabeto $\mathcal{A}$. Fissato $w \in \mathcal{C}$, la *sfera di Hamming* di centro w e raggio δ è l'insieme

$$B_\delta(w) = \{c \in \mathcal{C} : d(c, w) \leq \delta\}$$

Esempio 3.7. Sia $\mathcal{C} = \mathbb{R}^2$ uno spazio vettoriale di dimensione 2 sul campo reale e si consideri il punto $p = (p_1, p_2) \in \mathcal{C}$. Si osserva che

$$B_1(p) = \{(x, p_2) : x \in \mathbb{R}\} \cup \{(p_1, y) : y \in \mathbb{R}\}$$

mentre $B_2(p) = \mathbb{R}^2$

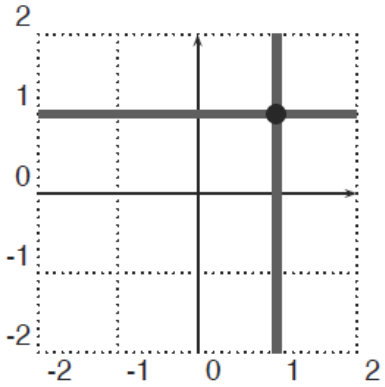


Figura 3: Sfera di Hamming in $\mathbb{R}^2$

3.3 Algoritmi di codifica e decodifica

Dato N il messaggio completo da inviare, si supponga di averlo diviso in pacchetti di k caratteri, o bit. Questi pacchetti vanno codificati, cioè vanno tradotti in una scrittura che va usata per la decodifica. Si chiami $\lambda : \mathcal{A}^k \to \mathcal{C}$ la funzione di codifica, con $\mathcal{C}$ un (n, M)-codice a blocchi per parole di lunghezza k. La funzione di codifica non fa altro che aggiungere dei bit al pacchetto, chiamati bit di ridondanza.

Definizione 3.8. La *ridondanza* di un (n, M)-codice finalizzato alla trasmissione di parole lunghezza k è la quantità

$$r = n - k$$

Questi hanno il compito, in fase di decodifica, di segnalare la presenza di eventuali errori di trasmissione causati dal canale ed eventualmente correggerli, laddove possibile.

Definizione 3.9. L'*efficienza* di un (n, M)-codice su un alfabeto $\mathcal{A}$ finalizzato alla trasmissione di parole di lunghezza k è la quantità $R := k/n$; se $M = \mathcal{A}^k$, ovvero se M è l'insieme di tutte le stringhe finite possibili di k elementi sull'alfabeto $\mathcal{A}$, e $|\mathcal{A}| = q$, allora si scrive

$$R = \frac{\log_q M}{n}$$

Esempio 3.10. Si supponga che il codice $\mathcal{C}$ di cui all'esempio 3.4 sia impiegato per trasmettere l'informazione rappresentata dalle coppie (00), (01), (10) e (11)

mediante la corrispondenza

$$(00) \longleftrightarrow (00000)$$
$$(01) \longleftrightarrow (10110)$$
$$(10) \longleftrightarrow (01011)$$
$$(11) \longleftrightarrow (11101)$$

Allora $R = \frac{2}{5}$ e $r = 3$

La funzione di decodifica è una funzione $\mu : \mathcal{A}^n \longrightarrow \mathcal{A}^k$. Ha come dominio l'intero insieme di tutti i messaggi possibili costituiti da n caratteri perchè deve poter valutare anche i messaggi ricevuti non correttamente. Per ogni blocco ricevuto m' di n caratteri, deve agire come segue:

1. Determinare la parola di codice c' più vicina a m'
2. Restituire $\mu(m') := \lambda^{-1}(c')$

Il seguente algoritmo descrive una prima strategia elementare che può essere implementata per cercare di decodificare un messaggio.

Algoritmo elementare di decodifica: dati un codice a blocchi $\mathcal{C}$ e un blocco m, bisogna determinare la distanza minima tra m ed una parola di codice $c \in \mathcal{C}$.
Sia $E = \{d(c, m) : c \in \mathcal{C}\}$, si ponga $e = \min E$. Se $B_e(m) \cap \mathcal{C}$ contiene un unico elemento x, si ponga $m' = x$ e si restituisca (e, m'): altrimenti si restituisca solamente e.

L'algoritmo richiede di calcolare M distanze distinte, per ogni operazione di decodifica. Il numero di operazioni da compiere è dunque una funzione lineare del numero di parole in $\mathcal{C}$.
E' possibile fare di meglio: l'algoritmo seguente fornisce esattamente gli stessi risultati del precedente, ma, in media, risulta molto più economico, in quando vengono calcolare le distanze indispensabili a fornire una risposta.

Algoritmo generico di decodifica (MDD): dati un codice a blocchi $\mathcal{C}$ e un blocco m, bisogna determinare la distanza minima tra m ed una parola di codice $c \in \mathcal{C}$.

1. Si ponga $e = 0$.

2. Se $D_e := B_e(m) \cap \mathcal{C} \neq \varnothing$ si distinguono due casi:

   a $|D_e| = 1$: allora si restituiscano e e l'unico elemento di D_e

   b $D_e > 1$: allora si restituisca solamente e

3. Se $D_e := B_e(m) \cap \mathcal{C} = \varnothing$, si ponga $e = e + 1$ e si ritorni al punto 2.

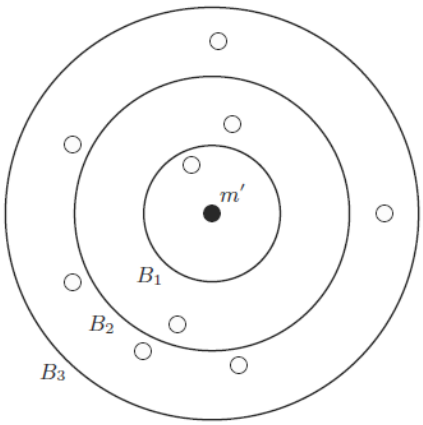


Figura 4: Algoritmo generico di codifica

Perchè ha senso procedere in tal modo? Perchè cercare la parola di codice che ha la minor distanza da quella ricevuta?
Si supponga di lavorare con un codice $\mathcal{C}$ di lunghezza n che usa un canale di trasmissione che ha probabilità di errore p per ogni bit inviato. Si riceve un blocco x che ha distanza di Hamming d da y. Allora la probabilità di aver ricevuto il blocco x al posto y è

$$\mathbb{P}(x \text{ anzichè } y) = p^d(1-p)^{n-d} = \left(\frac{p}{1-p}\right)^d (1-p)^n < (1-p)^n$$

Dunque la probabilità decresce al crescere di d. Pertanto la correzione degli errori con il principio della massima verosimiglianza, che coincide con il principio della minima distanza di Hamming, salvaguarda l'esigenza di minimizzare la probabilità di errore nella decodifica.
Il principio appena esposto si basa sull'assunzione che il numero di errori che possono avvenire in una trasmissione, sia tale da generare un blocco che non disti "troppo" dalla parola di codice corretta. Ciò significa che se in una comunicazione, un blocco dovesse subire eccessive modifiche, una volta arrivato potrebbe facilmente essere interpretato, con il principio di massima verosimiglianza, come un altra stringa, molto

diversa dall'originale. In quest'ultimo caso tale principio sarebbe completamente inutile, se non addirittura dannoso.
Se esiste una parola di codice $m' \in \mathcal{C}$ tale che $d(m, m') = e$, si dice che sono stati corretti e errori nel blocco ricevuto e la versione corretta è m'. Se, al contrario, vi sono più elementi di $\mathcal{C}$ a distanza e da m, si dice che sono stati individuati almeno e errori ma la correzione non è possibile. Si noti che se $\mathcal{C}$ è un codice di lunghezza n, tale algoritmo termina, dopo al massimo n passaggi.
Un modo elementare per risolvere il problema di calcolare gli insieme D_e, al variare di e fra 0 e n, è quello di enumerare tutte le parole del codice $\mathcal{C}$ e di controllare quali di esse soddisfano le condizioni richieste. Tale approccio risulta, però, decisamente dispendioso dal punto di vista computazionale; si rivela opportuno investigare i codici che posseggano una qualche struttura aggiuntiva grazie alla quale le procedure di codifica e decodifica possono essere rese notevolmente più veloci.

3.4 Distanza minima

Definizione 3.11. Sia $\mathcal{C}$ un (n, M)-codice. La *distanza minima* di $\mathcal{C}$ è la quantità

$$d = \min\{d(x, y) : x, y \in \mathcal{C},\ x \neq y\}.$$

Nel seguito si indicherà un (n, M)-codice $\mathcal{C}$ di distanza minima d con la dicitura (n, M, d)-codice.

Esempio 3.12. Si consideri un codice $\mathcal{C} = \{c_1, c_2, c_3, c_4, c_5\}$ sull'alfabeto $\mathcal{A} = \{a, b, c, d, e\}$ con

$$c_1 = (abcde) \quad c_2 = (acbed) \quad c_3 = (eabdc) \quad c_4 = (dceda) \quad c_5 = (adebc)$$

Per $i \neq j$ si ottiene $d(c_i, c_j) = 4$, pertanto tale codice ha distanza minima $d = 4$.

E' chiaro, per definizione di distanza di Hamming, che la distanza minima è sempre minore della lunghezza di un blocco, per cui $d \leq n$.
Per determinare la distanza minima d di un codice è necessario, a priori, calcolare la distanza fra $\binom{M}{2}$ coppie di parole.
Dato un (n, M)-codice $\mathcal{C}$ di distanza minima d, ogni due sfere di Hamming centrate in punti distinti di $\mathcal{C}$ e con raggio $r \leq \frac{d-1}{2}$ sono necessariamente disgiunte fra loro.

Definizione 3.13. Si dice che un codice $\mathcal{C}$ scopre e errori se, ogni qualvolta nella trasmissione di una parola vengono effettuati al più e errori, è possibile scoprirlo.

Definizione 3.14. Per ogni parola $w \in \mathcal{A}^n$, si definisce la distanza di w da $\mathcal{C}$ come

$$d(w, \mathcal{C}) := \min_{c \in \mathcal{C}} d(w, c)$$

Definizione 3.15. Si dice *raggio di copertura* di un codice $\mathcal{C} \subseteq \mathcal{A}^n$ il numero

$$\rho(\mathcal{C}) = \max_{x \in \mathcal{A}^n} d(x, \mathcal{C})$$

Esempio 3.16. Sia $\mathcal{C} = \{000, 111\}$. Facendo tutti i calcoli, il raggio di copertura è pari a 1. Se al destinatario arrivasse la stinga 001, si presume che il blocco sia arrivato incorretto, e quello giusto è in realtà 000, grazie al principio di massima verosimiglianza. Se però la stringa spedita fosse 000 e fossero modificati due bit, dando luogo al blocco, per esempio, 011, anche in questo caso il codice si accorgerebbe dell'errore, ma non lo correggerebbe adeguatamente.

Esempio 3.17. Sia ora $\mathcal{C} = \{c_1, c_2, c_3, c_4\}$ con

$$c_1 = (00000), \quad c_2 = (10110), \quad c_3 = (01011), \quad c_4 = (11101)$$

Il raggio di copertura, facendo i dovuti calcoli, è uguale a 2.
Si considerino tutte le sfere di raggio 1 centrate nelle parole di $\mathcal{C}$

$$\begin{aligned}
S_{c_1} &= \{(00000), (10000), (01000), (00100), (00010), (00001)\} \\
S_{c_2} &= \{(10110), (00110), (11110), (10010), (10100), (10111)\} \\
S_{c_3} &= \{(01011), (11011), (00011), (01111), (01001), (01010)\} \\
S_{c_4} &= \{(11101), (01101), (10101), (11001), (11111), (11100)\}
\end{aligned}$$

L'unione di tali sfere contiene 24 delle possibili 32 sequenze di lunghezza 5. Si indichi con S^* l'insieme delle sequenze che non cadono in alcuna delle sfere

$$S^* = \{(11000), (01100), (10001), (00101), (01110), (00111), (10011), (11010)\}$$

Queste sono tutte parole che hanno distanza di Hamming due dalle parole di codice. Se dovesse arrivare la stringa 00001, il codice si accorgerebbe dell'errore e il blocco verrebbe corretto in 00000. Se dovesse arrivare 11010, il codice sarebbe in grado di capire che ci sono stati due errori, ma non saprebbe individuarli.
Il raggio di copertura, dunque, quantifica il massimo numero di errori che il codice è in grado di rilevare: dato $e \leq \rho(\mathcal{C})$ il numero di errori avvenuti in una trasmissione, il codice è in grado di capire che ci sono stati e errori.
Se si riprende l'esempio precedente, nella trasmissione in cui avvengono due errori, il codice confonde un duplce errore con un errore singolo.

Definizione 3.18. Si dice che un codice $\mathcal{C}$ corregge e errori se, ogni qualvolta nella trasmissione di una parola vengono effettuati al più e errori, è possibile scoprirlo, ed è possibile correggere gli errori applicando il principio di massima somiglianza, in quanto esiste un'unica parola del codice che ha distanza di Hamming minima dal dato ricevuto.

Definizione 3.19. Il numero e di errori che un (n, M)-codice $\mathcal{C}$ garantisce di poter correggere, è detto *raggio di impacchettamento* di $\mathcal{C}$, ed è

$$\max_{i \in \mathbb{N}} \{B_i(c) \cap B_i(c') = \varnothing\} \ \forall \ c, c' \in \mathcal{C}.$$

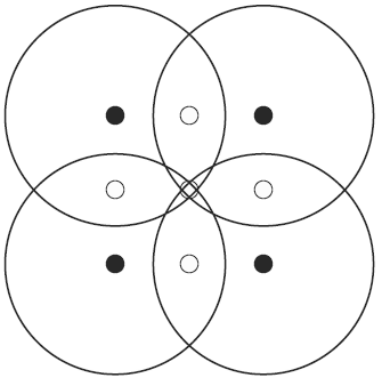


Figura 5: Raggio di copertura

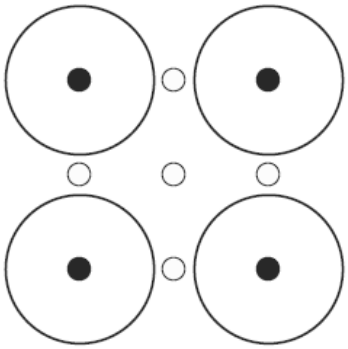


Figura 6: Raggio di impacchettamento

Definizione 3.20. Un codice per cui il raggio di copertura e quello di impacchettamento coincidono è detto *perfetto*.

In particolare, un (n, M)-codice $\mathcal{C}$ su un alfabeto $\mathcal{A}$ è perfetto se esso garantisce di poter correggere e errori e ogni parola di lunghezza n su $\mathcal{A}$ cade in una sfera di raggio e centrata in una parola di $\mathcal{C}$.
Il prossimo teorema collega il raggio di impacchettamento alla distanza minima del codice.

Teorema 3.21. *Sia $\mathcal{C}$ un (n, M, d)-codice. Allora:*

- *$\mathcal{C}$ scopre e errori $\iff d \geq e + 1$;*
- *$\mathcal{C}$ corregge e errori $\iff d \geq 2e + 1$;*

Dimostrazione. Se il codice $\mathcal{C}$ scopre e errori significa che è in grado di individuare una parola che differisce da una parola di codice di al massimo e bit. Le parole di codice dunque devo distare tra loro quindi almeno $e+1$, cosicchè non possa capitare che una parola subisca una modifica di e bit diventando un'altra parola di codice valida: dunque il codice $\mathcal{C}$ scopre e errori se e solo se le parole di codice distano tra loro almeno $e+1$. Ciò è equivalente a dire che la distanza minima del codice è almeno $e+1$.
Se il codice $\mathcal{C}$ corregge e errori vuol dire che è in grado di individuare e anche correggere una parola che differisce da una parola di codice di al massimo e. Poichè è in grado di individuarlo, da sopra, si ha che $d \geq e+1$. Poichè lo corregge anche, significa che la parola scorretta deve appartenere ad una e un'unica sfera di Hamming centrata in una parola valida. La distanza tra le parole quindi deve essere almeno $e+1$ più e affinchè la parola sbagliata finisca solo in una sfera di Hamming. Dunque $d \geq 2e+1$. □

Esempio 3.22. Si riprenda l'esempio 3.17. La distanza minima è $d = 3$.
Si riscrivano, per comodità, le sfere di raggio 1 e gli elementi non compresi in tali sfere:

$$S_{c_1} = \{(00000), (10000), (01000), (00100), (00010), (00001)\}$$
$$S_{c_2} = \{(10110), (00110), (11110), (10010), (10100), (10111)\}$$
$$S_{c_3} = \{(01011), (11011), (00011), (01111), (01001), (01010)\}$$
$$S_{c_4} = \{(11101), (01101), (10101), (11001), (11111), (11100)\}$$

$$S^* = \{(11000), (01100), (10001), (00101), (01110), (00111), (10011), (11010)\}$$

Per il teorema visto sopra è sempre possibile correggere al massimo un errore. Infatti, qualora un messaggio valido arrivasse con al più un bit sbagliato, rientrerebbe in una delle cinque sfere sopra riportate, e sarebbe facilmente correggibile, sempre in accordo con la massima verosimiglianza. In ogni altro caso, il codice non sarebbe in grado di correggere adeguatamente gli errori.
Si nota, infine, che qualora si sia a conoscenza che sul canale si possono verificare 3 o più alterazioni, non è possibile utilizzare il codice $\mathcal{C}$ nemmeno per sapere che ci sono stati degli errori, in quanto in $\mathcal{C}$ esiste una coppia di parole a distanza esattamente 3.

3.5 Limitazioni

I parametri n, M, d associati ad un codice a blocchi $\mathcal{C}$ su un alfabeto $\mathcal{A}$ non sono completamente indipendenti tra loro. Uno dei principali oggetti di studio nella teoria dei codici è formulare delle limitazioni che leghino tali parametri fra loro e, al contempo, trovare dei codici che siano ottimali dal punto di vista dei parametri. Chiaramente $M \leq |\mathcal{A}|^n$ e $d \leq n$.

Teorema 3.23. *(Limitazione di Singleton) Sia $\mathcal{C}$ un (n, M, d)-codice su un alfabeto $\mathcal{A}$. Allora*

$$M \leq |\mathcal{A}|^{n-d+1}$$

Dimostrazione. Si consideri l'insieme di tutte le M parole del codice $\mathcal{C}$ e si rimuovano da ognuna di esse gli ultimi $d-1$ simboli. Siccome due qualsiasi parole distinte di $\mathcal{C}$ differiscono di almeno d posizioni, le parole così ottenute sono ancora tutte diverse fra loro e dunque formano un $(n-d+1, M, d')$-codice su $\mathcal{A}$, con $d' \geq 1$. Ne segue che $M \leq |\mathcal{A}|^{n-d+1}$ □

Lemma 3.24. *Sia dato un codice di lunghezza n, su un insieme $\mathcal{A}$ di q elementi. Chiamato $|B_r(x)|$ il volume della sfera di Hamming centrata in x, vale*

$$|B_r(x)| = \sum_{i=0}^{r} \binom{n}{i} (q-1)^i \tag{6}$$

Dimostrazione. Dato l'insieme $\mathcal{A}^n$, questo ha in tutto q^n elementi. Vale banalmente

$$|B_n(x)| = q^n = ((q-1)+1)^n = \sum_{i=0}^{n} \binom{n}{i} (q-1)^i 1^{n-i} = \sum_{i=0}^{n} \binom{n}{i} (q-1)^i$$

Per r compreso tra 0 e n, la sfera di Hamming contiene i punti che stanno a distanza minore o uguale a r dal centro della sfera, ovvero i primi $r+1$ addendi della sommatoria. □

Teorema 3.25. *(Limitazione per impacchettamento di sfere) Sia $\mathcal{C} \subseteq \mathcal{A}^n$ un (n, M)-codice e-correttore e sia altresì $q = |\mathcal{A}|$. Allora,*

$$M \leq \frac{q^n}{\sum_{i=0}^{e} \binom{n}{i} (q-1)^i} \tag{7}$$

Dimostrazione. Per 3.21 si ha $d \geq 2e+1$, dunque per ogni coppia di punti $x, y \in \mathcal{C}$, le due sfere $B_e(x), B_e(y)$ sono disgiunte. D'altra parte, $\mathcal{A}^n$ contiene l'unione S di tutte le sfere $B_e(x)$ al variare di $x \in \mathcal{C}$, ed essendo tali sfere in numero di M e a due a due disgiunte, $|S| = M \cdot |B_e(x)|$ elementi, quindi

$$M \cdot |B_e(x)| \leq q^n$$

□

3.6 Codici equivalenti

In questo paragrafo si discute il problema di identificare quando due codici godono delle medesime proprietà e, pertanto, possono ritenersi equivalenti.

Definizione 3.26. Sia $\mathcal{A}$ un alfabeto e $n \geq 1$ un intero. Una applicazione $\varphi : \mathcal{A}^n \to \mathcal{A}^n$ è detta *isometria* di $\mathcal{A}^n$ se, per ogni $l, m \in \mathcal{A}^n$,

$$d(\varphi(l), \varphi(m)) = d(l, m)$$

In particolare, ogni isometria di $\mathcal{A}^n$ è una biiezione.

Definizione 3.27. Due codici $\mathcal{C}, \mathcal{C}'$ di lunghezza n su un alfabeto $\mathcal{A}$ sono detti *equivalenti* se è possibile ottenere $\mathcal{C}$ a partire da $\mathcal{C}'$ applicando le seguenti due operazioni:

- permutare le posizioni dei simboli in tutte le parole di $\mathcal{C}$
- permutare i caratteri in ogni singola posizione di $\mathcal{C}$

In generale, se due codici sono equivalenti, allora esiste una isometria di $\mathcal{A}^n$ che trasforma l'uno nell'altro.

Dimostrazione. Siano $\mathcal{C}$ e $\mathcal{C}'$ due codici equivalenti. Ciò significa, dalla definizione, che ogni parola di $\mathcal{C}'$ può essere ottenuta a partire da una parola c di $\mathcal{C}$ permutando le posizioni dei simboli nelle parole di $\mathcal{C}$ oppure permutando tutti i caratteri di ogni parola.
L'azione di permutare le posizioni dei simboli è equivalente a moltiplicare un vettore v che ha come elementi i simboli che compongono la parola c, per una matrice $n \times n$ di permutazione P.
Si pensa ora all'insieme $\mathcal{A}$ come ad un insieme ordinato. E' possibile pensare dunque

che se si intende prendere un elemento di $\mathcal{A}$ e trasformarlo in un altro elemento, bisognerà "spostarsi" in avanti nell'insieme ordinato fino a raggiungere il simbolo desiderato, facendo attenzione che quando si arriva all'ultimo elemento dell'insieme, procedendo in avanti, si ricomincia dal primo. Perciò, anche in questo caso, si possono vedere i blocchi c e c', elementi rispettivamente di $\mathcal{C}$ e $\mathcal{C}'$, come vettori v e v' di uno spazio n-dimensionale. Se si vuole quindi modificare la i-esima componente del vettore v e farla diventare dal k-esimo simbolo di $\mathcal{A}$ al $(k+j)$-esimo simbolo di $\mathcal{A}$, la funzione che rappresenta questa trasformazione è una traslazione di un vettore, la cui i-esima componente vale j.
La composizione di queste due azioni, viste come applicazioni su uno spazio affine, è un'isometria di $\mathcal{A}^n$. Infatti la permutazione di simboli per mezzo di una matrice di permutazione non modifica le distanze tra gli elementi; se due blocchi di $\mathcal{C}$ avevano una determinata distanza di Hamming, questa viene conservata anche in seguito alla trasformazione, per costruzione della distanza. Analogamente in seguito alla permutazione di singoli caratteri. □

Esempio 3.28. Si consideri un alfabeto $\mathcal{A} = \{a, b, c\}$. Sia $\mathcal{C}$ il $(3, 3)$-codice le cui parole sono

$$(aaa), (abc), (cbb).$$

Allora, il codice $\mathcal{C}'$ le cui parole sono

$$(aaa), (bac), (bcb)$$

è equivalente a $\mathcal{C}$, in quanto le sue parole sono state ottenute permutando le posizioni di ogni parola di $\mathcal{C}$. La permutazione è quella che scambia il primo e il secondo carattere di ogni parola.
Anche il codice $\mathbb{C}''$

$$(baa), (bbc), (abb)$$

è equivalente a $\mathcal{C}$. In questo caso le parole si sono ottenute sostituendo, nella prima posizione, il carattere b al carattere a, il carattere a al carattere c ed infine il carattere c al carattere b.
Il codice $\mathcal{C}'''$

$$(baa), (cac), (ccb)$$

è anch'esso equivalente a $\mathcal{C}$ ed è ottenuto effettuando entrambe le operazioni di cui sopra.
Per concludere, il codice $\mathcal{D}$

$$(aaa), (aab), (aca)$$

non è equivalente a nessuno di quelli visti sopra, in quanto non isometrico a nessuno di essi.

Per definizione di isometria, due codici equivalenti hanno le stesse proprietà metriche; pertanto distanza minima, raggio di copertura e raggio di impacchettamento coincidono. Inoltre, poichè φ è biiettiva su $\mathcal{A}^n$ anche il numero di parole dei codici è il medesimo e, pertanto, anche la ridondanza ed efficienza sono le stesse.

4 Codici lineari

Un codice correttore è costituito da due componenti base:

1. un insieme di parole su di un alfabeto
2. una corrispondenza fra tali parole e i messaggi che si voglio trasmettere

E' sempre possibile descrivere questi due elementi scrivendo per esteso l'insieme delle parole e la corrispondenza necessaria fra di esso e i messaggi; d'altro canto, questo approccio, tranne che nei casi più semplici, si rivela estremamente poco pratico.
In effetti, per poter concretamente implementare un codice correttore sono necessari tre elementi:

1. Un algoritmo di codifica;
2. Un algoritmo per individuare ed eventualmente correggere gli errori;
3. Un algoritmo di decodifica

Tutti e tre questi algoritmi devono, idealmente, essere efficienti, anche se, in funzione del tipo di applicazione considerata, è possibile talvolta accettare dei compromessi: in particolare, vi sono casi di comunicazione cosiddetta *asimmetrica*, in cui è importante che una delle procedure sia veloce, mentre le altre possono anche essere meno efficienti. Un esempio tipo è quello della televisione digitale, in cui l'apparato ricevente è molto meno potente da un punto di vista computazionale di quello trasmittente (per cui deve essere facile decodificare); un altro esempio è offerto dalle sonde interplanetarie, per cui la procedura di codifica deve essere efficiente, mentre la decodifica può richiedere anche molto tempo.
In questo capitolo si studieranno codici dotati di una struttura aggiuntiva di spazio vettoriale: i codici lineari. In particolare, si vedrà come sia possibile per essi fornire in modo semplice e conciso una descrizione degli algoritmi sopra indicati.

4.1 Generalità

L'alfabeto $\mathcal{A}$ su cui un codice correttore è definito è, a priori, solamente un insieme finito di simboli, senza alcuna ipotesi su sue ulteriori proprietà. Casi interessanti si verificano quando su $\mathcal{A}$ sono definite delle operazioni algebriche; in particolare, si rivela molto utile investigare il caso in cui $\mathcal{A}$ sia almeno un gruppo. In questo capitolo

e nei successivi si considererà una situazione più particolare, ma di fondamentale importanza: quella in cui $\mathcal{A}$ è un campo finito.
Nel seguito si supporrà sempre di avere un alfabeto $\mathcal{A} = \mathbb{F}_q$, dove $\mathbb{F}$ è un campo contenente esattamente q elementi. Sotto questa prima ipotesi, le parole di un codice $\mathcal{C}$ di lunghezza n si possono vedere come elementi dello spazio vettoriale $V_n(\mathbb{F}_{||}) = \mathbb{F}_q^n$, di dimensione n sul campo $\mathbb{F}_q$. Si supponga ora, in aggiunta, che l'insieme delle parole di codice $\mathcal{C}$ sia esso stesso un sottospazio vettoriale di dimensione k e che, al contempo, l'insieme K di tutti i possibili pacchetti da inviare, sia lo spazio vettoriale $V_k(\mathbb{F}_q)$ di dimensione $k \leq n$ sul campo $\mathbb{F}_q$. In questo modo, si ottengono codici contenenti $M = q^k$ parole e ogni codifica è descritta da un'applicazione iniettiva $\theta : V_k(\mathbb{F}_q) \to V_n(\mathbb{F}_q)$. La situazione più interessante si verifica quando θ è lineare.

Definizione 4.1. Un (n, M)-codice $\mathcal{C}$ è un $[n, k]$-codice *lineare* quando sono soddisfatte contemporaneamente le seguenti condizioni

- L'alfabeto su cui il codice è definito è un campo finito $\mathbb{F}_q$
- L'insieme K delle parole codificabili è uno spazio vettoriale di dimensione k sul campo $\mathbb{F}_q$ e, dunque, esso può identificarsi con $V_k(\mathbb{F}_q)$; in particolare, $M = q^k$
- L'operazione di codifica $\theta : V_k(\mathbb{F}_q) \to V_n(\mathbb{F}_q)$ è lineare ed è iniettiva.

Da qui in poi,il campo $\mathbb{F}_q$ sarà sempre e solo $\mathbb{Z}_q$. Per brevità, si userà il termine $[n, k, d]$-codice per indicare un $[n, k]$-codice lineare con distanza minima di Hamming d.

Esempio 4.2. Si costruisca un codice lineare $\mathcal{C}$ su $\mathbb{Z}_2$ di dimensione 2 e lunghezza 5. Sia $K = V_2(\mathbb{Z}_2)$ l'insieme di tutti i possibili pacchetti; pertanto, $K = \{(00), (10), (01), (11)\}$ e si indichi con $\mathcal{C}$ il sottospazio vettoriale di $V_5(\mathbb{Z}_2)$ avente per base

$$\mathscr{B} = \{(10111), (11110)\}$$

Si ottiene in modo naturale la seguente applicazione di codifica

$$(00) \mapsto (00000)$$
$$(10) \mapsto (10111)$$
$$(01) \mapsto (11110)$$
$$(11) \mapsto (01001)$$

Quindi

$$\mathcal{C} = \{(00000), (10111), (01001), (11110)\}$$

Una verifica diretta mostra come la distanza minima di $\mathcal{C}$ sia 2.

L'operazione di codifica di un messaggio mediante un codice lineare è una trasformazione lineare e, pertanto, risulta relativamente semplice da descrivere mediante una matrice.

4.2 Proprietà dei codici lineari

Le limitazioni viste nel precedente capitolo possono essere riscritte nel modo seguente per codici lineari

- **Limitazioni di Singleton:**

$$d \leq n - k + 1 \tag{8}$$

- **Limitazione per impacchettamento di sfere di Hamming**

$$\sum_{i=0}^{e} (q-1)^i \binom{n}{i} \leq q^{n-k} \tag{9}$$

Definizione 4.3. Un $[n, k, d]$-codice su $\mathbb{Z}_q$ si dice MDS (*maximum distance separable*) se i suoi parametri soddisfano l'uguaglianza nella limitazione di Singleton, ovvero $d = n - k + 1$

Esempio 4.4. Si consideri il codice lineare $\mathcal{C}$ su $\mathbb{Z}_5$ generato dai vettori

$$\mathscr{B} = \{(3410), (0341)\}$$

Questo codice ha parametri [4,2]. I suoi 25 elementi sono elencati qui sotto

(0000)	(0132)	(0214)	(0423)	(0341)
(1034)	(1111)	(1243)	(1402)	(1320)
(2013)	(2140)	(2222)	(2431)	(2304)
(4021)	(4103)	(4230)	(4444)	(4312)
(3042)	(3124)	(3201)	(3410)	(3333)

Un calcolo diretto mostra che la distanza minima di questo codice è 3. Poichè $3 = 4 - 2 + 1$, ne segue che $\mathcal{C}$ è un codice MDS.

Definizione 4.5. Il *peso di Hamming w(x)* di un vettore $x \in V_n(\mathbb{Z})$ è il numero di componenti di x diverse da 0

Definizione 4.6. Il *peso di Hamming* o *peso minimo w($\mathcal{C}$)* di un $[n, k]$-codice $\mathcal{C}$ è dato da

$$w(\mathcal{C}) = \min\{w(x) : x \in \mathcal{C}, x \neq 0\}$$

L'importanza del peso minimo di un codice lineare è dovuta al seguente teorema che mostra come esso coincida con la distanza minima.

Teorema 4.7. *Sia $\mathcal{C}$ un codice lineare. Per ogni coppia di parole $c, c' \in \mathcal{C}$ si ha*

$$d(c, c') = w(c - c')$$

Dimostrazione. La distanza fra due parole c, c' qualsiasi n $\mathcal{C}$ è la cardinalità dell'insieme $\Delta = \{i : c_i \neq c'_i\}$; chiaramente, $\Delta = \{i : c_i - c'_i \neq 0\}$. Dunque,

$$d(c, c') = d(c - c', 0) = w(c - c')$$

□

Corollario 4.8. *Sia d la distanza minima di un codice lineare $\mathcal{C}$; allora*

$$d = w(\mathcal{C})$$

Un'applicazione diretta della nozione di peso di una parola è data dagli insiemi di livello.

Definizione 4.9. Sia $\mathcal{C}$ un codice di lunghezza n su $\mathbb{Z}_q$. Per ogni intero $t \leq n$, l'*insieme di livello t* $\mathcal{W}_t(\mathcal{C})$ è

$$\mathcal{W}_t(\mathcal{C}) := \{c \in \mathcal{C} : w(c) = t\}$$

Per le considerazioni fatte nella dimostrazione del teorema precedente, la cardinalità dell'insieme $\mathcal{W}_t(\mathcal{C})$ coincide con il numero di parole di $\mathcal{C}$ che sono a distanza t da una qualsiasi parola $c \in \mathcal{C}$.

4.3 Matrice generatrice di un codice lineare

Gli elementi di un codice lineare $\mathcal{C}$ di dimensione k e lunghezza n costituiscono uno spazio vettoriale V, sottospazio di $V_n(\mathbb{Z}_q)$; pertanto, risulta possibile caratterizzarli fornendo una base di V rispetto una qualsiasi base prefissata di $V_n(\mathbb{Z}_q)$; solitamente, si supporrà che tale base preassegnata sia quella canonica $\mathscr{C}$. In questo modo, si riescono ad elencare le q^k parole di $\mathcal{C}$ a partire da un dato costituito da soli k vettori. Per definire completamente un codice serve anche un funzione di codifica.

Definizione 4.10. Una *matrice generatrice* di un $[n, k]$-codice lineare $\mathcal{C}$ è una matrice $k \times n$ le cui righe sono le componenti rispetto alla base canonica $\mathscr{C}$ di $V_n(\mathbb{Z}_q)$ dei vettori che formano una base di $\mathcal{C}$.

Esempio 4.11. I vettori $v_1 = (10001), v_2 = (11010), v_3 = (11101)$ formano una base per un $[5, 3]$-codice $\mathcal{C}$ binario. La matrice sarà

$$G = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Poichè $k = 3$, il pacchetto da codificare sarà lungo tre bit. Dato un vettore $m = (m_1, m_2, m_3) \in \mathbb{Z}_2^3$, rappresentante il pacchetto di dati da trasmettere, è possibile calcolare la parola di codice corrispondente

$$c = m \cdot G = m_1 v_1 + m_2 v_2 + m_3 v_3$$

Questo è il vettore che sarà trasmesso. Ad esempio, per $m = (101)$ si ha

$$m \cdot G = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

La distanza minima di questo codice, facendo i calcoli, è 2.

La base di uno spazio vettoriale non è unica; conseguentemente non lo è nemmeno la matrice generatrice di un codice lineare.

Esempio 4.12. Si generi ora un altro $[5, 3]$-codice mediante la base costituita dai vettori

$$u_1 = (10001) \quad u_2 = (01010) \quad u_3 = (00111)$$

In questo caso, il sottospazio 3-dimensionale di $V_5(\mathbb{Z}_2)$ ottenuto è diverso rispetto all'esempio precedente. In particolare, si ha

$$G = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Anche questo codice ha distanza minima $d = 2$

Se si applicano alle righe della matrice generatrice di un codice lineare $\mathcal{C}$ le seguenti tre operazioni:

1. moltiplicazione per uno scalare;
2. somma di un multiplo scalare di una differente riga;
3. permutazione;

si ottiene una nuova matrice che genera lo stesso codice lineare. Si noti, comunque, che il peso delle righe in due matrici generatrici distinte per lo stesso codice non è necessariamente lo stesso e che la codifica dei messaggi che si ottiene è differente. Nel seguito di questo paragrafo si descriverà una forma particolarmente comoda che può assumere la matrice generatrice di un codice: la cosiddetta *forma standard*.

Definizione 4.13. Una matrice generatrice G per un codice $\mathcal{C}$ è detta *matrice generatrice standard*, quando essa ha la forma

$$G = (I_k|A)$$

dove I_k denotala matrice identità $k \times k$ e A è una matrice $k \times (n-k)$

Un codice per cui è possibile fornire una matrice generatrice standard è detto *sistematico*. La matrice G dell'esempio 4.12 è in forma standard.

Definizione 4.14. Sia $\mathcal{C}$ un $[n,k]$-codice. Un insieme $I = \{i_1, ..., i_k\}$ di k posizioni è un *insieme di informazione* se, comunque dati $v_1, ..., v_k \in \mathbb{Z}_q$, esiste un'unica parola di codice c con $c_{i_1} = v_1, ..., c_{i_k} = v_k$

In particolare, un insieme I è di informazione per un codice $\mathcal{C}$ se, e soltanto se, le colonne indicizzate da I nella matrice generatrice di $\mathcal{C}$ sono fra loro linearmente indipendenti. Si vede immediatamente che un codice $\mathcal{C}$ è sistematico se, e soltanto se, l'insieme $I = \{1, 2, ..., k\}$ è di informazione per una sua matrice generatrice.

Esempio 4.15. Si consideri la codifica della parola $m = (101)$ secondo il codice G' dell'esempio 4.12; essa è $c = (10110)$. Si osserva che in questo caso una copia del vettore m compare nelle prime tre componenti di c; in effetti, tale proprietà vale per tutte le parole codificate mediante G' e risulta particolarmente vantaggiosa in sede di decodifica. Il codice dell'esempio 4.11, al contrario, non gode di tale proprietà.

Esempio 4.16. Sia $\mathcal{C}$ il $[7, 3, 3]$-codice lineare su $\mathbb{Z}_2$ con matrice generatrice

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

In questo caso l'insieme $I = \{1, 2, 3\}$ non è di informazione, per cui $\mathcal{C}$ non è un codice sistematico. Un possibile insieme di informazione per $\mathcal{C}$ è $I = \{5, 6, 7\}$.

Come visto in precedenza, un codice sistematico ammette una funzione di codifica tale che una copia del messaggio originario compare nelle prime k posizioni di ogni parola codificata, mentre le restanti $n - k$ posizioni forniscono ridondanza. I primi k bit identificati dagli indici appartenenti all'insieme di informazione, vengono detti bit di dati, per distinguerli da quelli di informazione.

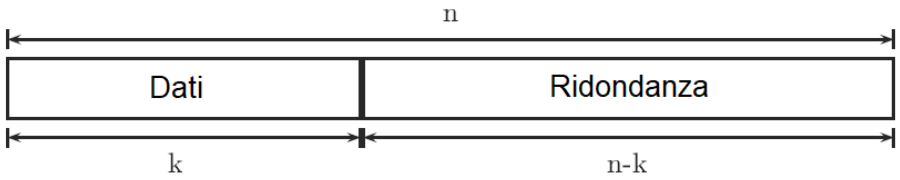


Figura 7: Codifica sistematica

E' importante in ogni caso ricordare che anche altri insiemi di k colonne oltre a $\{1, 2, ..., k\}$ potrebbero essere d'informazione, per cui la distinzione fra simboli di informazione e simboli di controllo è, in un certo senso, artificiale.

4.4 Ortogonalità e codice duale

Come visto nei paragrafi precedenti, l'insieme delle parole di un codice lineare costituisce un sottospazio di uno spazio vettoriale finito. In tali spazi vettoriali non si può introdurre la nozione di prodotto scalare in senso euclideo; è però ugualmente

possibile definire il concetto di ortogonalità fra vettori.
L'esempio di ortogonalità più importante che verrà considerato è l'applicazione

$$\langle \cdot, \cdot \rangle : \begin{cases} V_n(\mathbb{Z}_q) \times V_n(\mathbb{Z}_q) \to \mathbb{Z}_q \\ (f, g) \mapsto \sum_{i=1}^{n} f_i g_i \end{cases}$$

Definizione 4.17. Il *codice duale*, o *complemento ortogonale*, $\mathcal{C}^\perp$ di un $[n, k]$-codice $\mathcal{C}$ sopra un campo $\mathbb{Z}_q$, è il sottospazio vettoriale di $V_n(\mathbb{Z}_q)$

$$\mathcal{C}^\perp = \{x \in V_n(\mathbb{Z}_q) : \langle x, y \rangle = 0 \ \forall y \in \mathcal{C}\}$$

Teorema 4.18. *Se $\mathcal{C}$ è un $[n, k]$-codice sul campo $\mathbb{Z}$, allora $\mathcal{C}^\perp$ è un $[n, n-k]$-codice su $\mathbb{Z}$*

Dimostrazione. Si osserva che $\mathcal{C}^\perp$ è un insieme di elementi di $V_n(\mathbb{Z}_q)$ descritto da k condizioni lineari fra loro indipendenti; ne segue che la dimensione di $\mathcal{C}$ è proprio $n - k$. □

Diversamente da quanto succede in spazi vettoriali reali con il prodotto scalare ordinario, lo spazio vettoriale $\mathcal{C}^\perp$ non è, in generale, disgiunto da $\mathcal{C}$; in particolare, è possibile che si verifichi che $\mathcal{C}^\perp = \mathcal{C}$. In quest'ultimo caso si dice che il codice $\mathcal{C}$ è *autoduale*. Ogni vettore $c \in \mathcal{C}^\perp \cap \mathcal{C}$ è *isotropo*.
Data una matrice generatrice in forma standard per il codice $\mathcal{C}$, risulta facile scrivere una matrice generatrice per $\mathcal{C}^\perp$.

Teorema 4.19. *Sia $G = (I_k|A)$ è una matrice generatrice per il codice $\mathcal{C}$; allora*

$$H = (-A^T|I_{n-k})$$

è una matrice generatrice per il codice $\mathcal{C}^\perp$.

Dimostrazione. La matrice H contiene $n-k$ righe, fra loro linearmente indipendenti. Ne segue che il codice generato da H ha la medesima dimensione di $\mathcal{C}^\perp$. Si deve dunque solamente mostrare che ogni riga di H è un elemento di $\mathcal{C}^\perp$. A tal fine, sia m un pacchetto: la parola di codice corrispondente ad m, ovvero la codifica di m, è $c = mG$.
D'altro canto, per costruzione di H, si ha

$$GH^T = 0$$

da cui

$$cH^T = mGH^T = 0$$

per ogni parola di codice c. Ne segue che tutte le colonne di H^T corrispondono a vettori di $\mathcal{C}^\perp$. □

Esempio 4.20. Sia $\mathcal{C}$ il $[7, 4]$-codice su $\mathbb{Z}_3$ con matrice generatrice

$$G = \begin{pmatrix} 0 & 0 & 0 & 2 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 2 & 0 \\ 2 & 1 & 0 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Per trovare una matrice del tipo $G' = (I_4|A)$ che generi un codice $\mathcal{D}$ equivalente a $\mathcal{C}$, risulta necessario permutare le colonne di G, moltiplicando a destra per la matrice di permutazione

$$P = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

In tal modo si riesce a scrivere

$$G' = G \cdot P = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 0 & 2 & 2 & 2 \\ 0 & 0 & 0 & 1 & 2 & 1 & 0 \end{pmatrix} = (I_4|A), \qquad A = \begin{pmatrix} 0 & 2 & 1 \\ 0 & 1 & 2 \\ 2 & 2 & 2 \\ 2 & 1 & 0 \end{pmatrix}$$

Una matrice generatrice di $\mathcal{D}^\perp$ è

$$H' = (-A^T|I_3) = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 2 & 1 & 2 & 0 & 1 & 0 \\ 2 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

ripermutando le colonne di H', cioè moltiplicando a destra per la matrice di permutazione

$$P^T = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

si ottiene la seguente matrice generatrice per $\mathcal{C}^{\perp}$:

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 2 & 1 & 1 & 0 & 2 \\ 0 & 1 & 1 & 0 & 2 & 1 & 0 \end{pmatrix}$$

Si osserva che

$$GH^T = GI_7H^T = G(PP^T)H^T = (GP)(P^TH^T) = G'(HP)^T = G'H'^T = 0.$$

Inoltre, H ha, come previsto, rango $n - k = 3$.

5 Codici a rivelazione d'errore

Quando si ha a che fare con un canale di comunicazione, il problema sta nel creare un codice che possa compensare l'inevitabile rumore presente al suo interno. Quando la comunicazione è verbale, anche attraverso un canale di comunicazione come le linee telefoniche, tra due individui, se c'è presenza di errore è sufficiente chiedere all'interlocutore di ripetere il messaggio. Grazie alla complessità e alla conoscenza della lingua, il destinatario sarà in grado di comprendere il contenuto del messaggio, anche se è presente un errore nella seconda trasmissione.
Se la trasmissione avviene attraverso computer, sono presenti diverse limitazioni: l'elenco dei messaggi deve essere finito, le istruzioni inviate devono essere trasformate in sequenze finite non troppo lunghe, per rendere il sistema di comunicazione il più efficiente possibile, vi è continua presenza di disturbi, dovuti al canale di comunicazione che non può essere modificato...
Urgono dunque dei metodi per, in primo luogo, la rivelazione di errori all'interno dei messaggi, per evitare che un messaggio sia interpretato in maniera sbagliata. In secondo luogo c'è bisogno di procedure di correzione del messaggio compromesso, così che non debba essere necessario farselo rinviare.

5.1 Controllo di parità

Tale metodo viene implementato se il campo con cui si lavora è $\mathbb{Z}_2$. La maniera più semplice di codificare un messaggio binario per rendere possibile una rivelazione di un errore è contare il numero di 1 al suo interno, e aggiungere un bit, che viene denominato *bit di controllo di parità* o più generalmente *bit di parità*.
Dato un messaggio $m = (a_1, a_2, ..., a_{n-1})$ verrà codificato dalla seguente funzione

$$m = (a_1, a_2, ..., a_{n-1}) \mapsto m' = \left(a_1, a_2, ..., a_{n-1}, \sum_{i=1}^{n-1} a_i \right)$$

Andando così a generare un $[n, n-1]$-codice. Questo codice ha però grandissimi difetti, che lo rendono fondamentalmente inutilizzabile. Per rivelare un errore basterà sommare tutte le componenti del messaggio m' ricevuto, o alternativamente calcolare il suo peso. Se il risultato è un numero dispari il messaggio ha subito una variazione in uno dei suoi bit, ma non è possibile stabilire con certezza quale. Se il risultato è un numero pari, teoricamente non è avvenuta nessuna alterazione, in pratica dipende dal numero di alterazioni che potrebbe ricevere all'interno del canale

di comunicazione: se il canale, di norma, altera due bit, questo tipo di codifica non sarà in grado di rivelare l'errore. Allo stesso modo se ne avvengono quattro, sei, e così via.
Il metodo appena presentato, è in grado solo di rivelare un errore, e solo se il canale di comunicazione altera al più un bit ogni n.
E' necessario implementare dei metodi che riescano almeno a rivelare e correggere autonomamente almeno un errore. Per fare questo, sarà molto comoda la nozione di codice ortogonale.

Definizione 5.1. Sia $\mathcal{C}$ un $[n,k]$-codice su $\mathbb{Z}_q$. Si dice *matrice di controllo di parità* di $\mathcal{C}$ una matrice generatrice H di $\mathcal{C}^{\perp}$.

Osservazione. Un vettore v dello spazio vettoriale $V_n(\mathbb{Z}_q)$ è una parola di codice per $\mathcal{C}$ se, e soltanto se, $vH^T = 0$

Da qui discende l'utilità della matrice di controllo di parità H associata ad un $[n,k]$-codice $\mathcal{C}$.

Definizione 5.2. Sia H la matrice di controllo di parità associata ad un $[n,k]$-codice $\mathcal{C}$ sopra $\mathbb{Z}_q$. La *sindrome* di un vettore $x \in V_n(\mathbb{Z}_q)$ è il vettore $s \in V_{n-k}(\mathbb{Z}_q)$ ottenuto come $s = xH^T$

Per definizione di matrice di controllo di parità, le parole di codice sono tutti e soli i vettori di sindrome nulla. In particolare, un metodo per identificare eventuali errori di trasmissione è quello di calcolare la sindrome di ogni parola ricevuta e segnalare un errore se essa risulta diversa dal vettore nullo.
Si conclude con alcuni teoremi che mostrano come le proprietà strutturali della matrice di controllo di parità di $\mathcal{C}$ possano essere utilizzate per determinare una limitazione inferiore alla distanza minima dello stesso codice.

Teorema 5.3. *Il rango della matrice di controllo di parità H di un $[n,k,d]$-codice $\mathcal{C}$ è almeno $d-1$*

Dimostrazione. Si supponga che H abbia rango r, e si denoti con H_i la sua i-esima colonna. Allora, esistono degli indici $i_1, i_2, ..., i_{r+1}$ tali che

$$c_{i_1}H_{i_1} + c_{i_2}H_{i_2} + ... + c_{i_{r+1}}H_{i_{r+1}} = 0,$$

dove i $c_{i_1}, ..., c_{i_{r+1}}$ non sono tutti nulli e le restanti componenti di c sono tutte nulle. Ne segue che la parola c, che ha come componente i_j-esima c_{i_j}, appartiene sicuramente a $\mathcal{C}$ ed ha peso al massimo $r+1$; pertanto, $d \leq r+1$, da cui deriva il teorema. □

Vale anche una forma inversa per il teorema appena dimostrato.

Teorema 5.4. *Sia H la matrice di controllo di parità di un $[n, k]$-codice $\mathcal{C}$ e si supponga che ogni insieme di $r-1$ colonne di H sia linearmente indipendente. Allora, la distanza minima d del codice $\mathcal{C}$ è almeno r.*

Dimostrazione. Per ipotesi, ogni insieme di $r-1$ colonne di H è linearmente indipendente; ciò significa che, affinchè un vettore $c \neq 0$ abbia sindrome 0, e quindi appartenga al codice, esso deve contenere almeno r entrate non nulle. Ne segue che $d \geq r$. □

Teorema 5.5. *La minima distanza d di un $[n, k]$-codice avente matrice di controllo di parità H è il minimo ordine di un insieme linearmente dipendente di colonne della matrice H.*

Dimostrazione. Siano $h_1, ..., h_n$ le colonne di H e sia $x = (x_1, ..., x_n)$ una parola non nulla di $\mathcal{C}$. Si ha $x_1 h_i + ... + x_n h_n = 0$, ovvero $x \in \mathcal{C}$ determina una relazione di dipendenza lineare non banale tra le colonne di H e viceversa. Il peso di x, d'altra parte, determina il numero di colonne di H coinvolte in tale relazione. Da qui e da 4.8 segue la tesi. □

Esempio 5.6. Si consideri il $[4, 2]$-codice su $\mathbb{Z}_3$. Una sua matrice generatrice è

$$G = \begin{pmatrix} 2 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

Da questa, si può ricavare la matrice di controllo di parità corrispondente

$$H = \begin{pmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 2 & 2 \end{pmatrix}$$

Ogni coppia di colonne della matrice H è costituita da colonne indipendenti. Quindi $d = 3$ e il codice corregge un errore.

5.2 Codici di Hamming

In questa sezione si adotterà un approccio più algebrico, per trovare il miglior schema di codifica per un canale in cui si assume possa avvenire al massimo un singolo errore. Da qui in poi si assume sempre, a meno che non sia specificato, che il numero di errori che un canale di trasmissione può fare è minore o uguale al raggio di

impacchettamento del codice.
L'obiettivo è quello di leggere, successivamente alla trasmissione del messaggio, la *sindrome* del messaggio, ovvero una sequenza di bit dipendente dal messaggio ricevuto, che identifichi la presenza di un errore e sia in grado di correggerlo. Se la sindrome ha solo componenti nulle allora si presume che il messaggio sia arrivato senza modifiche. In caso contrario, le cifre diverse da 0 devono essere in grado di indicare la posizione nella quale è stato cambiato il bit del messaggio.
Perchè possa avvenire questo, il messaggio lungo n, deve soddisfare la seguente

$$2^r \geq n+1 \implies 2^r - 1 \geq n \tag{10}$$

Osservazione. Si nota che tale condizione equivale alla 9 con $q = 2$.

Dove r sarà il numero di bit della sequenza della sindrome. Questo perchè per ogni messaggio bisogna che ci sia una sequenza di bit che descriva l'esito "Non c'è stato alcun errore" e l'esito "C'è stato un errore correggibile al bit i" per ogni $i = 1, ..., n$. Per fare questo si necessita di un numero di bit r tale da poter descrivere ogni posizione del messaggio di n più uno, per poter dire che la sequenza è arrivata senza errori.
Il caso più facile da analizzare è quando viene soddisfatta l'uguaglianza in 10. Bisogna capire come deve essere fatto il messaggio di lunghezza n. Si vuole che la decodifica sia una sorta di funzione $\mathcal{A}^n \to \{0,1\}^r$, in maniera tale da associare ad ogni messaggio una sindrome che sarà in grado di esprimere in maniera univoca se e dove c'è stato un errore. Se gli n bit fossero tutti indipendenti tra loro, non si avrebbe modo di capire se e quando è stato alterato un bit, perchè bisognerebbe mandare, insieme al messaggio, la procedura per verificare che suddetto messaggio sia arrivato senza modifiche. Bisogna ideare una procedura che controlli, in maniera aritmetica, qualche proprietà che deve essere rispettata dai bit del messaggio per decretare se questo è corretto o no. Perciò, alcuni bit (di controllo) devono essere funzione dei restanti bit (di dati). Ma quanti? Chiaramente l'obiettivo è minimizzare il numero di bit di controllo per poter massimizzare la quantità di informazione inviata in un solo blocco.
Una maniera per creare una associazione tra sindrome e posizione è fare in modo che la sindrome esprima la posizione in espansione binaria; così facendo, la sindrome nulla non viene assegnata a nessuna posizione, indice del fatto che il messaggio è arrivato senza alterazioni, le altre sindromi vengono assegnate alle rispettive posizioni, secondo la tabella sottostante.

Posizione	Rappresentazione binaria
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
⋮	⋮

Detto questo, si immagini ora di prendere tutte le possibili sindromi e si prenda uno spazio costituito da 2^r punti, le sindromi, che va a costruire un ipercubo r-dimensionale, fatto nella seguente maniera: poichè i bit della sindrome sono r, si considerino gli insiemi $S_1, S_2, ..., S_r$, le r ipersuperfici che formano le facce del cubo, tali che nell'insieme S_i ci siano tutte e sole le sindromi che hanno l'$i$esimo bit uguale a 1. Ogni S_i è uno spazio affine di dimensione $r-1$, contiene 2^{r-1} sindromi, avendo noi fissato un bit. Il complementare è ovviamente sempre uno spazio affine di dimensione $r-1$, la faccia opposta dell'ipercubo.
Presi r iperpiani diversi, che siano S_i o S_i^c per qualche i, la loro intersezione sarà uguale ad una e unica sindrome: si verrebbe a creare un sistema di r equazioni, poichè ogni S_i può essere descritto da una equazione a r incognite, in r incognite. Ogni sindrome è univocamente identificata dall'appartenenza o meno agli iperpiani S_i.
Il messaggio dunque, al suo interno, deve contenere r bit che dicano se la sindrome generata dal messaggio appartiene a quali ipersuperfici. Dunque i bit di controllo di un messaggio lungo n devono essere r, con conseguenti $n - r = 2^r - r - 1 = k$ bit di dati. Si capisce che anche la quantità di messaggi differenti di cui si può disporre dipende in realtà dal numero di bit di controllo che si vuole inserire nel messaggio da inviare.

Definizione 5.7. Un $[2^r - 1, 2^r - 1 - r]$-codice è detto *codice di Hamming*.

Per facilitare il procedimento, i bit di controllo sono quelli in posizione 2^i con $i = 0, ..., r-1$. Quando viene codificato un messaggio, i suoi bit vengono traslati verso destra, rispettando sempre il loro ordine di scrittura, affinchè non occupino le

posizioni dedicate ai bit di controllo.
Ora bisogna solo capire come calcolare la sindrome in modo che questa indichi il bit scorretto. Un'idea è sfruttare il semplice controllo di parità. Poichè un messaggio corretto ha sindrome nulla, si scrivono le r relazioni, una per ogni bit di controllo, che un messaggio deve rispettare, partendo da quella riferita al primo bit di controllo. Guardando la tabella soprastante, ci si accorge che i numeri dispari hanno, nella loro rappresentazione binaria, un 1 come primo bit a destra; ovvero, hanno nella loro scrittura in base due, l'addendo 2^0. Come prima relazione, si inserisce il bit in maniera tale che la somma tra il bit in prima posizione e il bit in tutte le posizioni dispari faccia 0; alternativamente il primo bit di controllo è uguale alla somma dei bit in posizione dispari.
Per il secondo bit di controllo analogo ragionamento: è uguale alla somma di tutti i bit la cui posizione, in scrittura in base due, ha l'addendo 2^1. In generale, il j-esimo bit di controllo, che ricopre la posizione 2^{j-1}, sarà la somma di tutti i bit la cui posizione, in scrittura in base due, ha l'addendo 2^{j-1}.
Se nella trasmissione dovesse modificarsi il bit h-esimo, allora tutte le equazioni che lo coinvolgono darebbero risultato 1, e le altre risultato 0. Ogni equazione uguale a 1, genera un 1 nella sindrome in corrispondenza di ogni potenza di 2 presente nella scrittura in base due dell'indice h, andando così a descrivere univocamente tale posizione. Una volta saputa la posizione, è possibile correggere il suddetto bit.

Esempio 5.8. Si prenda un $[7, 4]$-codice: significa che i bit di controllo sono 3, da cui ogni messaggio che si vuole codificare è lungo quattro bit, per un totale da inviare di 7 bit. Per convenzione, i bit di controllo sono in posizione 2^i-esima con $i = 0, 1, ..., r - 1$. Dunque, per codificare il messaggio, si scrivono i 4 bit di informazione nelle posizioni 3,5,6 e 7, e i bit di controllo in posizione 1,2 e 4. Si supponga che il messaggio da codificare sia 1011. Questo viene trasformato nel messaggio –1-011 e gli spazi vuoti vengono riempiti dai bit di controllo.
Il controllo verrà attutato nella maniera seguente: poichè il primo bit, posto in posizione 1, come visto prima, indica le posizioni 1,3,5 e 7, il ricevitore del messaggio controllerà la somma di questi quattro bit, e ne scriverà l'esito sul posto più a destra di una tripla inizialmente lasciata bianca —. Analogamente farà lo stesso con gli altri due bit di controllo. Ogni 1 che appare nella tripla di controllo identifica un insieme di numeri e ogni 0 che appare identifica il suo complementare. L'intersezione dei tre insiemi avrà un solo elemento, che indicherà il bit dove è avvenuta la modifica.
Poichè lo scopo è creare un messaggio valido, il primo bit deve essere tale che la

somma con i bit in posizione 3, 5 e 7 faccia 0. Dunque il primo bit è 0. Con conti uguali si scrivono il secondo e il terzo bit di controllo. Il messaggio da inviare è quindi 0110011.
Una volta inviato il messaggio, al ricevente arriva il messaggio 0100011. Egli dunque controllo se è un messaggio valido, eseguendo l'algoritmo sopra descritto. La tripla che ne risulta è 011, quindi l'errore è al terzo bit e il messaggio che doveva arrivare è 0110011.
A questo punto il ricevente elimina i bit di controllo e legge il messaggio 1011.

L'esempio appena fatto è molto semplice, permette di codificare ed inviare 16 messaggi diversi (tutte le combinazioni di quattro bit). Per comprendere appieno la potenza di tale metodo, si immagini di voler creare un codice con più parole, partendo da 10 bit di controllo. Si crea così un $[2^{10}-1, 2^{10}-1-10] = [1023, 1013]$-codice, ossia parole da codificare lunghe 1013 bit, dunque 2^{1013} parole in totale!

Osservazione. Un codice di Hamming è un codice perfetto.

Esempio 5.9. Chiaramente è possibile applicare lo stesso procedimento a qualsiasi codice, che rispetta semplicemente la disuguaglianza (10). Si prende ad esempio una sequenza di otto bit, che si vuole codificare, per esempio 11011001. Si può notare che $8 \neq 2^r - 1 - r$ per qualsiasi r, dunque si costruirà un codice non ottimale, con $r = 4$, poichè si ricorda che l'obiettivo è rispettare la disuguaglianza inserendo comunque il minor numero di bit di controllo.
Il procedimento è identico a quello visto nel precedente esempio: la sequenza viene trasformata nella seguente –1-101-1001 e completata rispettando il seguente sistema

$$\begin{cases} \text{I bit di controllo} = 1+1+1+1+0 \mod 2 \\ \text{II bit di controllo} = 1+0+1+0+0 \mod 2 \\ \text{III bit di controllo} = 1+0+1+1 \mod 2 \\ \text{IV bit di controllo} = 1+0+0+1 \mod 2 \end{cases}$$

facendolo diventare quindi 001110101001. Il controllo e la decodifica funzionano analogamente a come già mostrato.

Una domanda che potrebbe essere lecito porsi è: "Quale deve essere al massimo, la probabilità p affinchè il codice di Hamming abbia una probabilità inferiore ad α di non poter correggere un dato messaggio?"
La domanda si traduce semplicemente nella seguente disequazione:

$$\mathbb{P}(\text{Ricevere un messaggio non correggibile}) \leq \alpha \ \Rightarrow$$

$$\Rightarrow\ 1 - \mathbb{P}(\text{Ricevere un messaggio correggibile}) \leq \alpha\ \Rightarrow$$

$$\Rightarrow\ 1 - \mathbb{P}(\text{Nessun bit corrotto}) - \mathbb{P}(\text{Un solo bit corrotto}) \leq \alpha\ \Rightarrow$$

$$\Rightarrow\ 1 - (1-p)^n - p(1-p)^{n-1}n \leq \alpha$$

Per evitare calcoli inutilmente complessi, può essere comodo fare un cambio di variabile

$$1 - q^n - (1-q)q^{n-1}n \leq \alpha$$

Si risolve l'equazione polinomiale seguente

$$1 - q^n - (1-q)q^{n-1}n - \alpha = 0 \Longrightarrow (n-1)q^n - nq^{n-1} + (1-\alpha) = 0$$

Prendendo in considerazione l'unica radice reale compresa tra zero e uno.
Tale radice indica la probabilità massima di modificare un unico bit di messaggio che deve avere il canale.

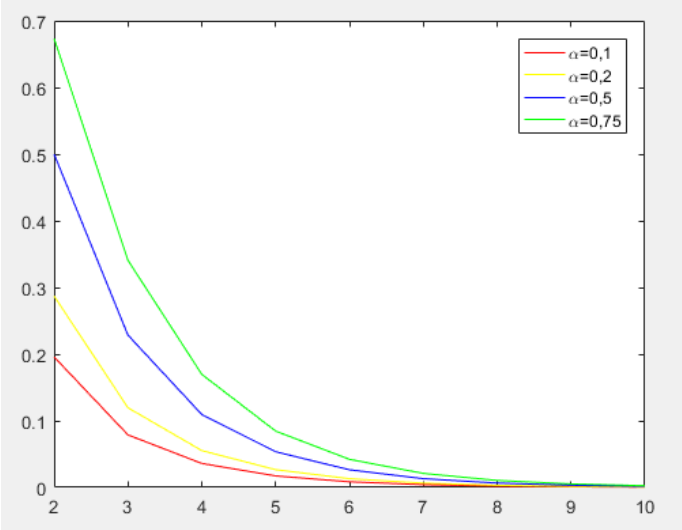


Figura 8: Andamento del valore massimo che può assumere p in funzione di r, affinchè la probabilità di non poter correggere un messaggio sia inferiore di un determinato α.

L'andamento dei grafici rispecchia le previsioni della teoria: più α è piccolo, più la probabilità p di corrompere un bit deve essere bassa; e più r aumenta, più p deve essere piccolo, affinchè la probabilità totale rimanga al di sotto di α.

5.3 Correzione singola e duplice rivelazione

Una volta presentato il metodo per la rivelazione e correzione di un singolo errore, è possibile implementare un algoritmo di decodifica in grado di correggere al più un errore e contemporaneamente rivelarne al più due. Unisce i codici di Hamming appena visti e il semplice controllo di parità. Chiaramente, per poter agire in tal modo, è necessario aumentare la distanza minima del codice, portandola da 3 a 4. Operativamente la costruzione è la seguente: partendo dall'ipotesi che nella trasmissione possano essere modificati al massimo due bit, si codifica il messaggio come visto prima, costruendo un messaggio di n bit da inviare, e si aggiunge un ultimo bit di controllo, tale che la somma dei primi n bit e dell'ultimo faccia 0. Al ricevente arriva il messaggio e si può trovare in una delle seguenti situazioni:

- il messaggio non ha subito nessun tipo di modifica, quindi sia la sindrome che la somma dei bit dovranno essere uguali a zero;
- è stato corrotto un solo bit. Se è stato corrotto il bit di parità, la sindrome sarà uguale a zero e la somma dei bit sarà uguale a 1. Se è stato modificato uno dei restanti bit, la sindrome e la somma sono diversi da zero;
- sono stati cambiati due bit, perciò la somma dei bit rimane zero, mentre la sindrome sarà diversa da zero. In questo caso non è possibile stabilire quali siano i bit corrotti, potrebbero essere uno il bit di parità e l'altro uno degli altri bit oppure entrambi potrebbero appartenere al messaggio originale.

Il tutto è riassunto nella seguente tabella:

Sindrome	Somma dei bit	Esito
0	0	Messaggio corretto
0	1	Errore singolo in posizione $n+1$
non 0	1	Errore singolo in una delle prime n posizioni
non 0	0	Duplice errore

Osservazione. Nel terzo caso, se tutti e due i bit modificati appartengono al messaggio originale, la sindrome non può essere uguale a zero. Se così fosse, significherebbe che allo stesso codice appartengono due elementi che hanno distanza di Hamming tra loro di 2. Ma poichè si suppone che il codice corregga almeno un errore, la distanza tra le parole di codice deve essere almeno 3, in accordo con il teorema 3.21.

6 Teoria algebrica dei codici

Tutto quello visto in precedenza può essere formalizzato dal punto di vista algebrico. L'obiettivo di quest'ultimo capitolo è costruire la matrice di controllo di parità e quindi il codice costruendo un'analogia tra i blocchi e i polinomi. In questa trattazione si assume che il campo con cui si lavora è sempre $\mathbb{Z}_2$.
Prima di iniziare si ricordano alcuni concetti chiave senza i quali si rischia di non comprendere a pieno la trattazione:

- un *messaggio* è l'intera sequenza di informazioni che si vuole trasmettere al destinatario: può essere un testo, un capitolo di un libro, qualsiasi cosa;
- un *pacchetto* è una porzione di messaggio, trasformata in una stringa di elementi appartenenti a $\mathbb{Z}_q^k$;
- un *blocco* è un pacchetto codificato, quindi trasformato in una stringa di $\mathbb{Z}_q^n$.

Si riprenda il codice visto prima che si premurava solo di individuare un singolo errore, con ogni parola codificata ed inviata lunga n bit. Dato che la procedura di controllo consiste nel sommare tutti i bit del blocco, la sua matrice di controllo di parità è la seguente

$$H = (1, 1, 1, ..., 1)$$

con componenti tutte uguali a 1, con tante componenti quante sono quelle del blocco da controllare.
Si consideri un vettore m di n componenti tale che non abbia subito nessuna modifica all'interno del canale. Se si esegue la moltiplicazione tra c e H^T si ottiene

$$mH^T = 0$$

per ogni parola inalterata che viene inviata. Ciò è la spiegazione algebrica di quello che accade con il controllo semplice di parità: la moltiplicazione genera una somma delle componenti del vettore che genera uno 0 oppure un 1.
Si supponga che il vettore m subisca una modifica; questa converte uno dei bit. Il messaggio che arriva è dunque della forma

$$m + e$$

dove e rappresenta il vettore con tutti 0 e un 1 in corrispondenza della posizione dell'errore.

Se si procede come prima, si ottiene

$$(m+e)H^T = mH^T + eH^T = eH^T = 1 \neq 0$$

indice della presenza di un errore all'interno del messaggio.
Se però il messaggio subisce due modifiche, si ottiene

$$(m+e+e')H^T = mH^T + eH^T + e'H^T = eH^T + e'H^T = 1 + 1 = 0$$

come visto prima, il metodo non permettere di rivelare due errori, o quattro, o comunque un numero pari.

6.1 Rivisitazione di Hamming

Si riprenda ora il codice di Hamming come presentato nel paragrafo 5.2. Di seguito, si ricorda la corrispondenza tra posizione del bit di parità e le posizioni del vettore che dipendono da tale bit

1	1,3,5,7,9,11,13,15,...
2	2,3,6,7,10,11,14,15,...
3	4,5,6,7,12,13,14,15,...
4	8,9,10,11,12,13,14,15,...
	eccetera

Se si riprende l'esempio 5.8, bisogna costruire una matrice che controlli la somma tra la prima, la terza, la quinta e la settima componente; contemporaneamente deve controllare la somma tra la seconda, terza, sesta e settima, e infine la somma tra la quarta, quinta, sesta e settiman componente. Si ottiene la matrice

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{11}$$

dove le righe indicano le corrispondenze coi bit di controllo dal basso verso l'alto;
Sia m un messaggio inviato al destinatario. Come prima, se il messaggio non ha subito alterazioni, si avrà

$$mH^T = 0$$

Nuovamente, se il messaggio ha subito una singola alterazione, avrà forma

$$m + e_i$$

dove e_i è il vettore con tutti 0 e un 1 in posizione i-esima.
Se si prova a verificare se il messaggio ricevuto è corretto, si ottiene

$$(m + e_i)H^T = mH^T + e_iH^T = e_iH^T = H_i \neq 0$$

dove con H_i si intende la i-esima colonna di H. Infatti il codice di Hamming è in grado non solo di rivelare l'errore, ma anche di correggerlo, individuando la posizione della colonna corrispondente alla posizione dove è avvenuto l'errore.
Da questo semplice esempio si evince come non sia importante quali sono i bit di controllo, poichè la sindrome indica univocamente la colonna della matrice indipendentemente dalla scelta dei bit usati per il controllo.
E' possibile costruire un'analogia con il metodo di correzione singola e individuazione di un duplice errore; la strategia è sempre la stessa. Se si immagina riprendere lo stesso codice con $n = 7$, la matrice che risulta è la seguente

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

In seguito alla moltiplicazione tra un messaggio ricevuto, di n componenti, e tale matrice, si possono distinguere gli stessi casi visti prima

Prima componente	Restanti 7 componenti	Esito
0	nulle	Messaggio corretto
1	nulle	Errore singolo in posizione $n + 1$
1	non nulle	Errore singolo in una delle prime n posizioni
0	non nulle	Duplice errore

6.2 Polinomi e vettori

Un passaggio cruciale, che permette di dare dei metodi costruttivi che si possono implementare anche su calcolatori, è non pensare più a vettori ma a polinomi. Preso un blocco lungo n, si vuole pensare che i suoi bit siano i coefficienti, in ordine decrescente di grado da sinistra a destra, di un polinomio di grado $n - 1$.
Dato ed esempio, la sequenza

$$1,\ 0,\ 1,\ 1,\ 0$$

il suo polinomio associato è

$$P_1(x) = x^4 + x^2 + x$$

L'obiettivo, in un certo senso, è usare una funzione generatrice per rappresentare il codice.
Lavorando con insiemi di polinomi, possiamo sommare e moltiplicare tra loro elementi del codice sfruttando le proprietà dell'anello dei polinomi. Dato un codice lungo $2^r - 1$ bit, lo scopo è cercare un metodo per partire da un polinomio e generare tutte le colonne della matrice di controllo di parità, creando la corrispondenza tra colonne e i bit del messaggio. Per farlo, è necessario ricorrere ad anelli di polinomio quozientati con polinomi con gradi ben precisi. Per fare in modo che venga preservata, prima tra tutte, la legge di annullamento del prodotto, bisogna quozientare per un polinomio irriducibile. Bisogna solo capire, al variare di k, quali sono i polinomi irriducibili che si possono usare.

6.2.1 Polinomi irriducibili

Un polinomio irriducibile è semplicemente un polinomio monico che non può essere fattorizzato in polinomi di grado positivo inferiore. Poichè però si sta lavorando in $\mathbb{Z}_2$ è bene provare a vedere alcuni esempi di polinomio irriducibili tutt'altro che banali.
Per polinomi di grado zero l'unico esempio è il polinomio costante

$$P = 1$$

Per polinomi di grado 1, ci sono i seguenti

$$x, \quad x + 1$$

entrambi irriducibili per definizione.
Per il grado due bisogna cominciare ad andare per tentativi se si vuole compiere una fattorizzazione. D'altra parte, lavorando in un campo a due elementi, un buon modo per escludere quelli che sicuramente non sono irriducibili è controllare se 1 o 0 sono radici del polinomio da esaminare. Fatto questo, l'unico polinomio di grado 2 che potrebbe essere irriducibile è

$$x^2 + x + 1$$

Dato che questo, se fosse riducibile, si scomporrebbe in due polinomi di primo grado, cosa che non accade, esso è di fatto l'unico di secondo grado irriducibile.
Per il grado tre, ragionando come sopra, i due candidati sono

$$x^3 + x + 1, \quad x^3 + x^2 + 1$$

Anche questi due, se fossero riducibili, avrebbero come componenti un polinomio di primo grado. Quindi sono gli unici due polinomi irriducibili di terzo grado.
Per ultimi, solo a titolo di esempio, i polinomi di quarto grado, candidati ad essere irriducibili

$$x^4 + x + 1, \quad x^4 + x^2 + 1, \quad x^4 + x^3 + 1, \quad x^4 + x^3 + x^2 + x + 1$$

Nessuno di questi ha come soluzione 0 o 1, dunque non possono essere scomposti nel prodotto di un polinomio di primo e uno di terzo grado.
Essendo polinomi di quarto grado, potrebbero scomporsi in due componenti irriducibili di secondo grado. Poichè c'è un solo polinomio irriducibile di secondo grado, si nota che

$$(x^2 + x + 1)^2 = x^4 + x^2 + 1$$

dunque gli unici polinomi irriducibili di quarto grado sono

$$x^4 + x + 1, \quad x^4 + x^3 + 1, \quad x^4 + x^3 + x^2 + x + 1$$

Ora si può procedere alla costruzione di qualsiasi codice di Hamming.

6.2.2 Il [7,4]-codice di Hamming

Tale codice ha $2^4 = 16$ parole di codice, ognuno lungo $2^3 - 1 = 7$ bit. Per provare a costruire il codice, si parta da un polinomio di grado $k = 4 - 1 = 3$, per esempio $x^3 + x + 1$. Sia $a \in \mathbb{C}$ la radice di tale polinomio, tale che $a^3 + a + 1 = 0$; dunque $a^3 = a + 1$. Da qui si evince che ogni potenza di a può essere scritta come combinazione lineare di $1, a, a^2$.
Si considerino dei vettori colonna di tre componenti, che hanno componenti $1, a$ e a^2, scritti dal basso verso l'alto. Ogni vettore colonna esprime i coefficienti della corrispondente potenza di a

$$a^0 = 1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \qquad a = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \qquad a^2 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

Si rappresentino ora le altre potenze di a

$$a^3 = a + 1 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \qquad a^4 = a \cdot a^3 = a \cdot (a+1) = a^2 + a = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

$$a^5 = a \cdot a^4 = a \cdot (a^2 + a) = a^3 + a^2 = a^2 + a + 1 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$$a^6 = a \cdot a^5 = a \cdot (a^2 + a + 1) = a^3 + a^2 + a = a^2 + a + a + 1 = a^2 + 1 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

Si nota facilmente che $a^7 = 1$, dunque che a è una radice che genera tutto il codice. Se si scrivono i vettori colonna in una matrice, si ottiene

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

indicanti le potenze di a da destra a sinistra, dalla più piccola alla più grande. Tale matrice è, evidentemente, equivalente alla (11), secondo la relazione sottostante

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Se si osserva bene, la modifica delle colonne cambia anche il modo di leggere la sindrome. Questa non indica più la posizione in scrittura binaria di dove è avvenuto l'errore: ora l'errore viene trovato leggendo la sindrome come colonna di H. La sindrome, a meno che non sia nulla, è uguale alla colonna della matrice di controllo nella posizione nella quale è avvenuto l'errore.
Qualora si volesse inviare un messaggio, in analogia a quanto vista in precedenza, i primi quattro bit costituiscono i bit di dati e gli ultimi tre i bit di controllo.

A questo punto, come avviene la decodifica? Dato che lo scopo è ottenere un messaggio che, se non subisce alterazioni, dà come sindrome il vettore nullo, bisogna richiedere che il suo polinomio corrispondente sia divisibile per il polinomio generatore del codice, in questo caso $x^3 + x + 1$. Siano $c_1, ..., c_7$ i bit, ordinati da sinistra a destra, del blocco c da decodificare. Come detto sopra, ad ogni sequenza si può associare un polinomio, $p(x)$), i cui coefficienti sono proprio $c_1, ..., c_7$, dal grado più alto al più basso.
Se $p(x)$ è divisibile per $x^3 + x + 1$, significa che $p(x) = g(x)(x^3 + x + 1)$, per un qualche $g(x) \in \mathbb{Z}_2[x]$. Allora

$$0 = g(a)(a^3 + a + 1) = p(a) = c_1a^6 + c_2a^5 + ... + c_6a + c_7 =$$

$$= c_1(a^2 + 1) + c_2(a^2 + a + 1) + c_3(a^2 + a) + c_4(a + 1) + c_5a^2 + c_6a + c_7 =$$

Se ora si vede a come il vettore (010), la riga sopra si può riscrivere come

$$= c_1(101) + c_2(111) + c_3(110) + c_4(011) + c_5(100) + c_6(010) + c_7(001) =$$

$$(c_1, c_2, c_3, c_4, c_5, c_6, c_7) \cdot H^T = c \cdot H^T$$

Come fatto prima, si costruisce un messaggio di sette bit, riempiendo, stavolta, le prime quattro posizioni con bit di dati e riempiendo con 0 i tre bit di controllo, compiendo una divisione tra il polinomio associato al vettore costituito dai bit di dati per il polinomio generatore. Il resto ottenuto va aggiunto al vettore di dati, ottenendo così il messaggio codificato da inviare. Per la decodifica il procedimento è identico: si divide il polinomio associato al messaggio ricevuto per il polinomio generatore; il resto indica la colonna della matrice di controllo e quindi il bit alterato.

Esempio 6.1. Si immagini di voler codificare il pacchetto 1001. Dunque inizialmente si trasforma nella sequenza 1001000, che corrisponde al polinomio $x^6 + x^3$. Questo si divide per il polinomio $x^3 + x + 1$ e se ne calcola il resto.

$$x^6 + x^3 = (x^3 + x)(x^3 + x + 1) + (x^2 + x)$$

Dato il resto $x^2 + x$, associato alla sequenza 110, il pacchetto viene infine codificato nella sequenza da inviare 1001110, corrispondente al polinomio $x^6 + x^3 + x^2 + x$, divisibile per il polinomio generatore.
Si supponga ora che, durante la trasmissione, sia stato alterato il primo bit a sinistra, e dunque il blocco ricevuto sia 0001110. Si compie la divisione

$$x^3 + x^2 + x = 1(x^3 + x + 1) + x^2 + 1$$

generando il resto associato alla sequenza 101, indicante la prima colonna di sinistra della matrice di controllo.

Costruita la matrice di controllo a partire da un polinomio, si calcola la matrice generatrice del codice e si esegue codifica, decodifica ed eventuale correzione solo attraverso matrici.

Esempio 6.2. Data

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

come sopra, la matrice generatrice del codice associata è

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

Se il pacchetto è, per esempio, 1001, questo viene codificato, diventando

$$(1001) \cdot G = (1001110)$$

esattamente come visto sopra con i polinomi. Anche qui, si supponga che il blocco ricevuto sia 0001110. La sindrome si ricava dalla moltiplicazione tra il blocco ricevuto e la matrice di controllo:

$$(0001110) \cdot H^t = (101)$$

che corrisponde alla prima colonna di H. Dunque l'errore è avvenuto nel primo bit da sinistra. Una volta corretto, il blocco viene decodificato, passando da 1001110 a 1001 e infine letto dal destinatario.

6.3 Codici a correzione d'errore multipla

La correzione multipla sfrutta una relazione simile alla (10). Dati sempre r bit di controllo, se il codice corregge fino ad e errori, si vuole che le combinazioni di r bit siano in grado di indicare che sono stato commessi zero errori, che ne sia stato commesso uno solo e la sua posizione, che ne siano stati commessi due e le loro posizioni fino a e errori e la loro posizione. Dunque, deve valere la seguente:

$$\sum_{i=0}^{e} \binom{n}{i} \leq 2^r \tag{12}$$

Grazie a questa relazione, e stabilito il numero massimo di errori che il codice può correggere, si può procedere in due modi complementari; si decide quanto deve valere n e calcolare l'r più piccolo che soddisfa la disuguaglianza: r deve essere più piccolo possibile per poter inviare più dati possibili. Oppure decidere a priori il numero di bit di ridondanza e calcolare n più grande possibile, per la stessa motivazione di prima. Chiaramente, la lunghezza del blocco, n, dovrà essere strettamente maggiore del numero r di bit di ridondanza, altrimenti si invia un blocco senza alcun bit di dati.
Una volta stabiliti i parametri, si può creare un codice di Hamming analogo al $[7, 4]$-codice, con matrice di controllo, in cui ogni combinazione di r bit, quindi ogni sindrome possibile, esprime una possibilità di errore.
E' interessante notare che, posto $r = 4$, ad esempio, si vanno a generare due tipi di codice:

- Un $[15, 11]$-codice, in grado di correggere un solo errore. I dati trasportati da ogni messaggio sono undici;
- Un $[5, 1]$-codice, in grado di correggere due errori, costruito attraverso la 12, che trasporta un solo bit di dati;

Il $[5, 1]$-codice è senza dubbio molto inusuale, poichè ha poca utilità spedire un blocco di 5 bit dove solamente uno è di dati. Se si vuole costruire un codice che corregge fino a due errori si userà una ridondanza più alta per poter trasmettere più informazione.

Esempio 6.3. Si consideri il $[5, 1]$-codice. Al suo interno ci sono solo due parole. Queste sono il risultato della codifica di due parole di 1 bit ciascuno: 0 e 1. Perciò si può pensare, senza perdita di generalità, che le uniche due parole di codice siano 00000 e 11111. Per la definizione 4.10, la matrice generatrice del codice

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

La distanza minima d è pari a 5, dunque il codice è in grado di correggere fino a due errori. La matrice di controllo, secondo la 4.19, è

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Qualora dovesse verificarsi nella trasmissione un singolo errore, questo sarebbe immediatamente individuabile grazie alla sindrome, esattamente come visto prima.
Se dovessero esserci due bit sbagliati, la sindrome risultante sarebbe la somma delle due colonne indicanti ognuna il bit dove è avvenuto l'errore. Si pensi di voler inviare il messaggio 00000 e che questo venga modificato in 00101. La sindrome risultante è

$$00101 \cdot H^T = 0101$$

La sindrome è uguale alla somma della prima e terza colonna da destra, esattamente corrispondenti ai bit errati.

6.3.1 Tutto questo quanto costa?

Un aspetto che potrebbe essere molto interessante analizzare è studiare un codice di Hamming in relazione alla probabilità che ha ogni bit inviato di essere modificato dal canale di trasmissione. Se si prende in considerazione un codice di Hamming con r generico, è chiaro che la sua potenza è determinata dalla sua capacità di saper individuare e correggere un eventuale errore di trasmissione.
Si prenda in considerazione un messaggio lungo N, che si vuole inviare ad un ipotetico ricevitore; ad esempio il primo canto della Divina Commedia. Si vuole dividere un messaggio in tanti pacchetti e sfruttare un codice di Hamming per inviare più informazione possibile per pacchetto tenendo conto anche della possibilità che il canale possa modificare più bit di quanto il codice è in grado di correggere. Data lunghezza del pacchetto di dati di k bit, questa grandezza dipende strettamente da r. Il numero di pacchetti da inviare sarà, banalmente, $N/k(r)$ (qui non è importante che la la lunghezza complessiva del messaggio sia perfettamente divisibile per il numero di bit di dati).
Si provi a costruire una funzione che indichi il danno causato da un blocco irrimediabilmente corrotto e che tenga anche conto del costo per inviare ogni pacchetto.
Chiamata q la probabilità che arrivi un blocco non correggibile, si chiami D una funzione che indica il danno causato dal fatto di aver ricevuto un blocco non correggibile. Sia poi C_2 il costo per inviare un singolo blocco ed infine, n la lunghezza del blocco da inviare. La funzione è

$$f(n, r, e) = \frac{N}{k(r)} q(n, e) D(k) + C_2 \frac{N}{k(r)} n$$

La variabile q, dipende dal raggio di impacchettamento del codice e da n, dato che la probabilità di corruzione dipende da quanto è lungo il pacchetto. D dipende da k, e

dunque da r, intuitivamente, perchè è più dannoso ricevere un pacchetto illeggibile se questo è più lungo. Quest'ultima si può scrivere più semplicemente: una costante C_0, che simboleggia il danno causato dal solo fatto di aver ricevuto un blocco non correggibile, e una costante per una potenza di k, che indica un danno proporzionale alla grandezza della quantità di dati persa per via della corruzione:

$$f(N, n, r, e, a, C_0, C_1, C_2) = \frac{N}{k} q(C_0 + C_1 k^a) + C_2 \frac{N}{k} n = \frac{N}{k}(q(C_0 + C_1 k^a) + C_2 n) \quad (13)$$

Per semplicità si può studiare la funzione senza la presenza della costante N: moltiplica tutti i fattori è non modifica la presenza di eventuali presenza di minimo o concavità:

$$h(n, r, e, a, C_0, C_1, C_2) = \frac{1}{k}(q(C_0 + C_1 k^a) + C_2 n)$$

Il parametro a è un numero reale, che indica l'importanza della lunghezza del blocco perso in relazione al danno compiuto: più a si avvicina a 0, meno è importante se il blocco perso è lungo o corto, ogni stringa persa crea il medesimo danno costante, C_0; quindi per assurdo, con a vicino a -1, si cerca di usare molta ridondanza per poter trasportare più quantità di dati possibile. Si ricorda che quando si intende "ridondanza alta", si sta parlando del parametro r: più r aumenta, più il rapporto r/n tende a diventare zero e, di conseguenza, il rapporto k/n tende a 1. Più a cresce, più importanza viene data al blocco perso di maggior lunghezza.
Sia X la variabile aleatoria che esprime il numero di bit modificati nella trasmissione. Allora vale la seguente

$$\mathbb{P}(X = j) = \binom{n}{j} p^j (1-p)^{n-j}.$$

Dove p è la probabilità di modificare uno e un solo bit del messaggio.
Si pensi ora ad un codice di Hamming con raggio di impacchettamento e, ovvero che sia in grado di accorgersi e di correggere fino ad e errori. Sia E l'evento "Il blocco ricevuto è arrivato sbagliato e non correttamente correggibile". Perciò si può scrivere

$$\mathbb{P}(E) = \sum_{j=e+1}^{n} \binom{n}{j} p^j (1-p)^{n-j} = 1 - \sum_{j=0}^{e} \binom{n}{j} p^j (1-p)^{n-j} = q(n, e)$$

Esprime la probabilità che avvengano più di e errori, fissato p, e che quindi il codice di Hamming risulti inutile. E' una funzione che cresce al crescere di n; in effetti, al crescere di n, il blocco diventa sempre più lungo, diventando così più soggetto ad

errori non correggibili.
D'altro canto, bisogna tenere conto che un blocco più lungo trasporta più dati, quindi bisogna cercare di trovare un compromesso sul numero di bit di ridondanza, r, da usare.
L'intento è quello di riuscire a trasportare più bit di dati possibili. Ogni volta che viene deciso e è necessario ricalcolare il numero n che soddisfa la (12). Si definisce la seguente

$$g(r,e) = \max_{n>r} \left(\sum_{i=0}^{e} \binom{n}{i} \leq 2^r \right)$$

Ricordandosi che $k = n - r$, se ad ogni elemento di (13) si sostituisce la sua esplicita scrittura, si può studiare la funzione andando ad osservare la presenza di punti di minimo:

$$h(n,r,e,a,C_0,C_1,C_2) = \frac{1}{g(r,e)-r}(q(C_0 + C_1(g(r,e)-r)^a) + C_2 g(r,e))$$

E' doveroso ricordare che deve valere la disuguaglianza $n > r$, perchè si vuole che ci sia almeno un bit di dati nel blocco. Per questo motivo, una volta fissato il raggio di impacchettamento, n troppo piccoli non rispettano la (12), dunque vanno presi in esame valori di r da un certo punto in poi.

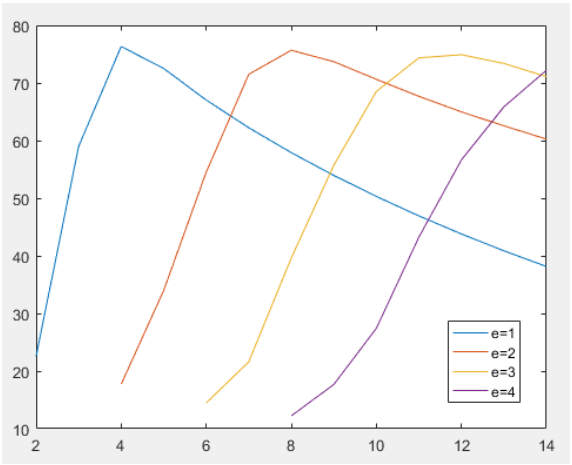


Figura 9: Grafico di h con $a = 0,9$, $C_0 = 1$ $C_1 = 100$ e $C_2 = 0,25$

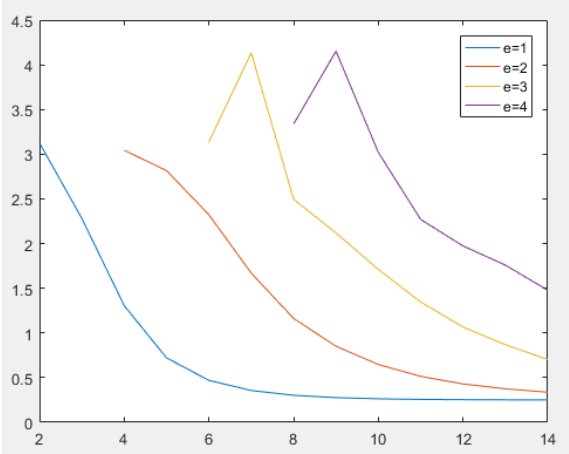


Figura 10: Grafico di h con $a = 0$, $C_0 = 1$ $C_1 = 10$ e $C_2 = 0,25$

In questo caso, con $a < 1$, la funzione tende al valore di C_2, quindi $0,25$. Per valori di r piccoli la funzione vale più di C_2, perciò, per ottenere valori bassi di h, ai fini pratici si usa un valore di r più alto possibile. Il caso limite, se $a = 0$, significa che un blocco grande o uno piccolo, se persi, creano lo stesso danno. Perciò è preferibile usare una ridondanza più elevata per trasportare più quantità di dati, quindi un rapporto k/n più vicino a 1 possibile.

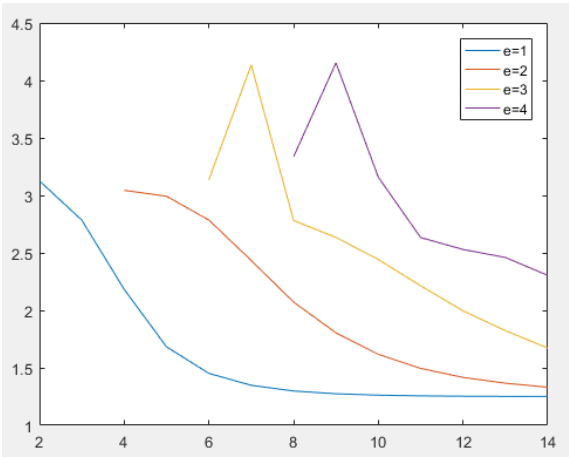


Figura 11: Grafico di h con $a = 1$, $C_0 = 10$ $C_1 = 1$ e $C_2 = 0,25$

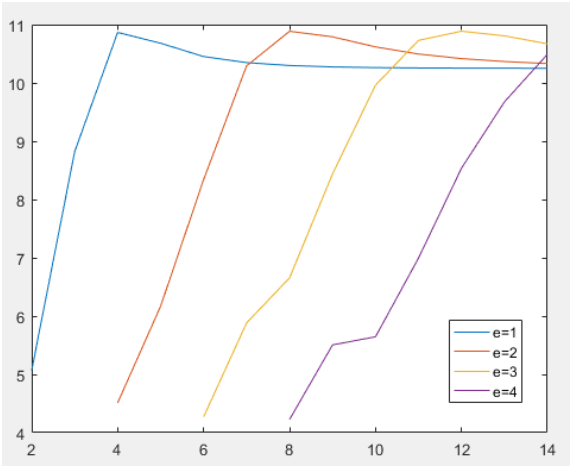


Figura 12: Grafico di h con $a = 1$, $C_0 = 10$ $C_1 = 10$ e $C_2 = 0,25$

Con $a = 1$, la parte di funzione che indica il danno tende al valore C_1, mentre la seconda parte, asintoticamente, tende sempre a C_2. In questo particolare caso, il minimo dipende unicamente dalla scelta delle due costanti. Con C_1 molto più grande di C_2 è preferibile usare ridondanza bassa. In effetti, se il danno nel perdere un pacchetto è molto più alto del costo per usare una data lunghezza nei blocchi, si cercherà di usare valori di r, bit di ridondanza, più piccoli.

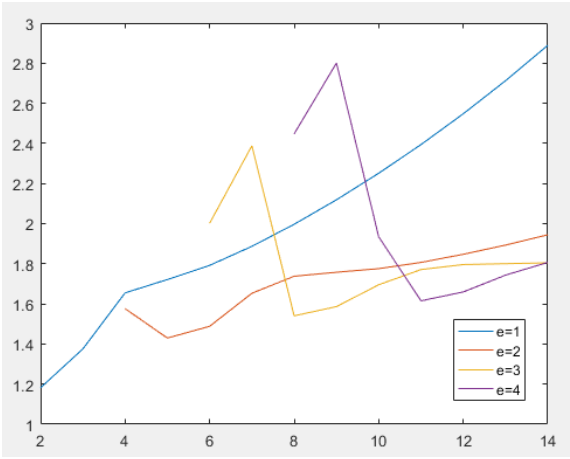


Figura 13: Grafico di h con $a = 1,1$, $C_0 = C_1 = 1$ e $C_2 = 0,25$

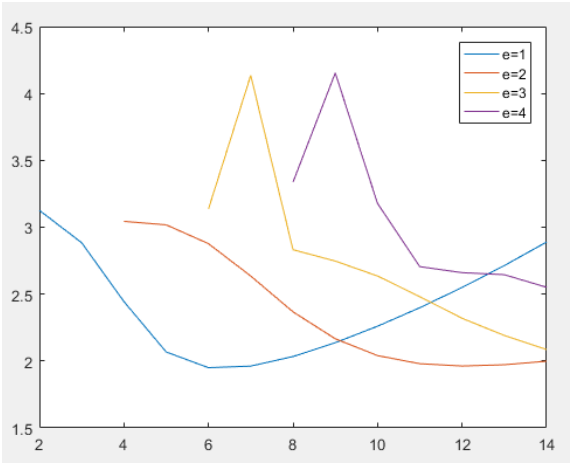


Figura 14: Grafico di h con $a = 1,1$, $C_0 = 10$ $C_1 = 1$ e $C_2 = 0,25$

In questi due grafici la funzione che calcola il danno causato da un blocco irrimediabilmente corrotto, asintoticamente, si comporta come una potenza positiva di k. Percio, per r che tende all'infinito, la funzione tenderà all'infinito. Si possono vedere chiaramente dei punti di minimo. Questa particolare combinazione di dati mostra come non sia sempre immediata la scelta del numero di bit di ridondanza r da usare per minimizzare la funzione.

6.4 Codici di Hamming con q qualsiasi

Come ultima cosa, i codici di Hamming si possono generalizzare con q primo qualsiasi (ricordiamo che q deve essere primo affinchè $\mathbb{Z}_q$ sia un campo).
Anche in questo caso, si può fare un ragionamento molto simile a quando si è costruito il codice di Hamming con $q = 2$. Si ritorni ora al caso in cui si vuole rilevare e correggere al massimo un errore. Dato q, se si dispone di r bit di ridondanza, si dispone di $q^r - 1$ combinazioni differenti che descrivono le diverse posizioni del blocco (ricordando sempre che c'è bisogno di una combinazione che indichi che il blocco è arrivato correttamente, quella tutta nulla).
Con $q = 2$, essendoci solo 1 e 0, era facile stabilire l'alternativa ad un bit sbagliato. Qui però, a differenza di prima, una volta individuato il bit scorretto, non è possibile correggerlo con certezza. Bisogna dunque sfruttare le $q^r - 1$ combinazioni e dare al loro interno delle informazioni che dicano non solo qual è il bit sbagliato, ma anche quale bit dovrebbe diventare.
Siccome le combinazioni possibili sono appunto $q^r - 1$ e ognuna di questa deve dire,

dato un bit errato, in quale dei restanti $q-1$ bit bisogna correggerlo, si ricava la definizione di un codice di Hamming in generale.

Definizione 6.4. Dato $n = (q^r - 1)/(q-1)$, il *codice di Hamming* di ordine r su $\mathbb{Z}_q$ è un $[n, n-r]$-codice la cui matrice di controllo di parità è una matrice $r \times n$ tale che due colonne qualsiasi non sono una il multiplo dell'altra.

Osservazione. Se si prende $q = 2$ si ottiene il codice di Hamming visto all'inizio del capitolo.

Anche in questo caso si procede tramite la matrice di controllo di parità, con una piccola costruzione che facilità la correzione.
Si supponga che un $[n, k]$-codice sia assegnato mediante una matrice di controllo di parità H. Si prendano in considerazione i vettori di $\mathbb{Z}_q^n$ che hanno peso 0 e 1. Questi sono esattamente q^r: l'unico elemento di peso 0 è il vettore 0^n, i restanti vettori di peso 1 sono $(q-1)$ perchè ci sono $(q-1)$ numeri diversi da zero, per ogni posizione del blocco, dunque n:

$$1 + (q-1)n = q^r$$

Si chiami l'insieme che contiene tali vettori, l'insieme dei *leader*. Dato v uno qualunque di questi vettori, se si somma v a tutte le parole del codice, si ottiene un insieme, indicato con $v + \mathcal{C}$, che si dice un *laterale* di $\mathcal{C}$.
Dato il codice $\mathcal{C}$, il laterale generato dalla somma con un vettore preserva la distanza tra le componenti del codice, poichè se due elementi distano tra loro d, in seguito alla somma con lo stesso vettore è variato lo stesso numero di componenti di entrambi i vettori.
Si nota che i laterali di $\mathcal{C}$ sono a due a due disgiunti e in corrispondenza biunivoca con $\mathcal{C}$, dunque contengono ciascuno q^k elementi. Se ci fosse un elemento w di $\mathbb{Z}_q^n$ che appartiene a due laterali differenti, supponiamo $v_1 + \mathcal{C}$ e $v_2 + \mathcal{C}$, significa allora che sia il vettore $w - v_1$ che il vettore $w - v_2$ appartengono a $\mathcal{C}$. Ma questi due elementi hanno distanza di Hamming tra loro pari a 2 o a 1. Allora il codice, non avendo distanza minima $d \geq 3$ non sarebbe in grado di correggere un singolo errore, andando contro l'ipotesi di essere un codice correttore.
Infine si osserva che tutti e soli i vettori nel laterale $v + \mathcal{C}$ hanno la stessa sindrome $v \cdot H^T$.
Si fa una lista di tutti i leader e delle relative sindromi, tutte possibili sindromi di elementi di $\mathbb{Z}_q^n$. Se si riceve un blocco x che ha sindrome non nulla, c'è un errore. Guardando la tabella dei leader, si trova il leader l avente la stessa sindrome di

x e si decodifica x sottraendogli il leader in questione. In tal modo si ottiene un elemento $y = x - l$ con sindrome nulla, dunque appartenente al codice. Inoltre, per la stessa definizione di leader come elemento di peso minimo nel suo laterale, y è un elemento di $\mathcal{C}$ avente distanza di Hamming minima da x. Dunque la decodifica avviene rispettando il principio di massima verosimiglianza.

Esempio 6.5. Si consideri il $[4, 2]$-codice su $\mathbb{Z}_3$. Una sua matrice generatrice è

$$G = \begin{pmatrix} 2 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

Da questa, si può ricavare la matrice di controllo di parità corrispondente

$$H = \begin{pmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 2 & 2 \end{pmatrix}$$

Essendo un campo con tre elementi e i bit di informazione due, le sindromi possibili, e di conseguenza i leader associati, sono i seguenti nove

(0,0,0,0)	(1,0,0,0)	(2,0,0,0)	(0,1,0,0)	(0,2,0,0)
(0,0)	(1,0)	(2,0)	(0,1)	(0,2)

(0,0,1,0)	(0,0,2,0)	(0,0,0,1)	(0,0,0,2)
(1,2)	(2,1)	(2,2)	(1,1)

Se si riceve la sequenza (2,1,2,1), la sua sindrome è

$$(2, 1, 2, 1) \cdot H^T = (0, 1)$$

Il relativo leader è (0,1,0,0), dunque si decodifica (2,1,2,1) come

$$(2, 1, 2, 1) - (0, 1, 0, 0) = (2, 0, 2, 1)$$

che è il messaggio originale che sarebbe dovuto arrivare.

Per concludere tutta la trattazione, dato un codice lineare, con base q qualsiasi, che corregge e errori, deve rispettare

$$\sum_{i=0}^{e} (q-1)^i \binom{n}{i} \leq q^r$$

che corrisponde alla (9). La spiegazione è che, come visto prima, gli r bit di ridondanza devono essere in grado di descrivere le seguenti situazioni;

- Nessun errore, dunque una combinazione di r bit;
- Un singolo errore, quindi $n \cdot (q-1)$ combinazioni di r bit, che devono descrivere dove è avvenuto l'errore e qual è il bit corretto;
- Un duplice errore, $\binom{n}{2} \cdot (q-1)^2$ combinazioni di r bit, che devono descrivere dove sono avvenuti gli errori e, per ogni coppia, indicare quali sono i bit corretti;

e così via.

7 Conclusioni

Dopo aver osservato e appreso come avviene sia la codifica-decodifica che la correzione, è naturale unire questi due aspetti per una trasmissione veloce ed efficiente. Le applicazioni di queste scoperte sono alla base delle odierne comunicazioni e telecomunicazioni: dalle radio ai telefoni, passando dagli apparecchi audiovisivi, costituiscono un elemento vitale per la socializzazione come è intesa oggi, senza la quale non esisterebbe nè il concetto nè la realtà della globalizzazione.
Assieme alla crittografia, la teoria dell'informazione e quella dei codici ci permette di vivere in un mondo che comunica, in maniera sicura e libera.

Riferimenti bibliografici

[1] Emanuele Angeleri. *Trasmissione dati*. Editoriale Delfino Milano, 1972.

[2] *Bit matematici, la teoria dell'informazione e delle telecomunicazioni*. Mondo matematico. RBA, 2018.

[3] Thomas J. A. Cover Th. M. *Elements of information theory*. Wiley-Interscience, 2006.

[4] *Esistono problemi irrisolvibili? Matematica, complessità e computazione*. Mondo matematico. RBA, 2018.

[5] R.W. Hamming. *Coding and Information Theory*. Prentice-Hall, 1986. ISBN: 0-13-139072-4.

[6] Marco Lenci. *Appunti del corso di Teoria dell'Informazione e Comunicazione per gli studenti del Corso di Laurea Magistrale, Curriculum Generale Applicativo*. 2020.

[7] Giuzzi Luca. *Codici correttori, un'introduzione*. Springer, 2006. ISBN: 1088-470-0539-6.

[8] G.M. Piacentini Cattaneo W.M. Baldoni C. Ciliberto. *Aritmetica, crittografia e codici*. Springer, 2006. ISBN: 1088-470-0455-1.