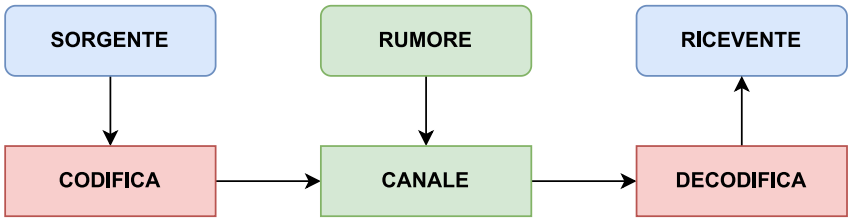
Teoria dell'Informazione e della Trasmissione

Indice

Introduzione

Lo schema di riferimento che andremo ad usare durante tutto il corso è il seguente:



La prima persona che lavorò alla teoria dell'informazione fu **Claude Shannon**, un impiegato della TNT (*Telecom americana*) al quale è stato commissionato un lavoro: massimizzare la quantità di dati che potevano essere trasmessi sul canale minimizzando il numero di errori che potevano accadere durante la trasmissione.

Nel 1948 Shannon pubblica un lavoro intitolato *«A mathematical theory of comunication»*, un risultato molto teorico nel quale modella in maniera astratta il canale e capisce come l'informazione può essere spedita *«meglio»* se rispetta certe caratteristiche. Ci troviamo quindi di fronte ad un risultato che non ci dice cosa fare nel caso specifico o che codice è meglio, ma è probabilistico, ti dice nel caso medio cosa succede.

Questo approccio è sicuramente ottimale, ma rappresenta un problema: va bene il caso medio, ma a me piacerebbe sapere cosa succede nel caso reale. Questo approccio più reale è quello invece seguito dal russo **Kolmogorov**, un accademico che vuole capire cosa succede nei singoli casi senza usare la probabilità. A metà degli anni '60 propone la sua idea di teoria dell'informazione, focalizzandosi quindi sui casi reali e non sui casi medi.

Questi due mostri sacri della teoria dell'informazione, nel nostro schema di lavoro, si posizionano nei rettangoli di sorgente e codifica, mentre la teoria della trasmissione si concentra sui rettangoli sottostanti, quindi nella parte di canale.

Altri due personaggi che hanno lavorato allo stesso problema di Kolmogorov sono due russi, che lavoravano uno in America e uno nell'est del mondo, ma non sono ricordati perché Kolmogorov era il più famoso dei tre.

Un altro personaggio importante è **Richard Hamming**, un ricercatore di Bell Lab che doveva risolvere un problema: i job mandati in esecuzione dalle code batch delle macchine aziendali se si piantavano durante il weekend potevano far perdere un sacco di tempo. La domanda che si poneva Hamming era *«che maroni, perché se le schede forate hanno errori, e le macchine lo sanno, io non posso essere in grado di correggerli?»*

Lui sarà il primo che, per risolvere un problema pratico, costruisce il primo codice di rilevazione e correzione degli errori, il famosissimo **codice di Hamming**.

Vediamo alcune date che rivelano quanto è stata importante la teoria dell'informazione:

- 1965: prima foto di Marte in bianco e nero grande $\approx 240K$ bit, inviata con velocità 6 bit al secondo, ci metteva ore per arrivare a destinazione;
- 1969: stessa foto ma compressa, inviata con velocità $16K$ bit al secondo, ci metteva pochi secondi;
- 1976: prima foto di Marte a colori da parte di Viking;
- 1979: prima foto di Giove e delle sue lune a colori da parte di Voyager;
- anni '80: prima foto di Saturno e delle sue lune da parte di Voyager.

Parte I — Teoria dell'Informazione

1. Codici

Sia X un insieme di **simboli** finito. Un/a **messaggio/parola**

$$\overline{x} = x_1 \cdot ... \cdot x_n \in X^n$$

è una concatenazione di n caratteri x_i di X. Dato $d > 1$, definiamo

$$D = \{0, ..., d-1\}$$

l'insieme delle **parole di codice**. Definiamo infine

$$c : X \longrightarrow D^+$$

la **funzione di codifica**, la quale associa ad ogni carattere di X una parola di codice.

Non useremo tanto D, ci sarà più utile

$$D^+ = \bigcup_{n \geq 1}^{\infty} \{0, ..., d-1\}^n$$

insieme di tutte le parole ottenute concatenando parole di D.

Voglio **massimizzare la compressione**: sia $l_c(x)$ la lunghezza della parola $x \in X$ nel codice c. Ci servirà anche la **probabilità** $p(x)$ di estrarre un simbolo: questo perché alle parole estratte spesso andremo a dare una lunghezza breve, mentre alle parole estratte di rado andremo a dare una lunghezza maggiore. Un codice che segue queste convenzioni è ad esempio il **codice morse**.

Definiamo il **modello sorgente** come la coppia $\langle X, p \rangle$.

Il nostro modello è quasi completo, perché ci interessa la probabilità di ottenere una parola di codice di D^+, non la probabilità dei simboli presenti in D. Definiamo quindi

$$P_n(\overline{x} = x_1 \cdot ... \cdot x_n) = \prod_{i=1}^{n} p(x_i).$$

Qua vediamo una prima enorme semplificazione di Shannon: stiamo assumendo l'**indipendenza** tra più estrazioni consecutive, cosa che nelle lingue parlate ad esempio non è vera.

Dati il modello $\langle X, p \rangle$, la base $d > 1$ e il codice $c : X \longrightarrow D^+$ il modello deve essere tale che

$$\mathbb{E}[l_c] = \sum_{x \in X} l_c(x) p(x)$$

sia minimo.

Il primo problema che incontriamo sta nella fase di decodifica: se codifico due simboli con la stessa parola di codice come faccio in fase di decodifica a capire quale sia il simbolo di partenza?

Definizione 1.1 (*Codice non singolare*)**:** Un codice c è non singolare se c è una funzione iniettiva.

Imponiamo che il codice c sia **non singolare**. Stiamo imponendo quindi che il codominio sia abbastanza capiente da tenere almeno tutti i simboli di X.

Definiamo $C : X^+ \longrightarrow D^+$ l'**estensione del codice** c che permette di codificare una parola intera. Se il codice c è non singolare, cosa possiamo dire di C?

Purtroppo, la non singolarità **non** si trasmette nell'estensione del codice.

Definizione 1.2 (*Codice univocamente decodificabile*): Un codice c è univocamente decodificabile se la sua estensione C è non singolare.

Restringiamo l'insieme dei codici solo a quelli UD. Per dimostrare che un codice è UD esiste l'algoritmo di **Sardinas-Patterson**, che lavora in tempo

$$O(mL) \quad | \quad m = |X| \wedge L = \sum_{x \in X} l_c(x).$$

Come funziona l'algoritmo di Sardinas-Patterson?

Sardinas-Patterson

input
- insieme S_1 contenente le parole del codice c

1: for $i = 1, 2, \ldots$
2: - Costruiamo l'insieme S_{i+1}
3: - - for $x \in S_1$
4: - - - Se $xy \in S_i$ allora $y \in S_{i+1}$
5: - - for $x \in S_i$
6: - - - Se $xy \in S_1$ allora $y \in S_{i+1}$
7: - Casi di terminazione:
8: - - Se $S_{i+1} = \varnothing$ allora c è UD
9: - - Se S_{i+1} è uguale ad almeno un insieme $S_j \mid j < i + 1$ allora c è UD
10: - - Se $S_{i+1} \cap S_1 \neq \varnothing$ allora c non è UD
11: - Se siamo arrivati fin qua ricominciamo il ciclo for con un nuovo valore di i

Con gli UD abbiamo un altro piccolo problema: non sono **stream**. In poche parole, un codice UD non ci garantisce di decodificare istantaneamente una parola di codice in un carattere di X quando mi arrivano un po' di bit, ma dovrei prima ricevere tutta la stringa e poi decodificare. Sono ottimi codici eh, però ogni tanto aspetteremo tutta la codifica prima di poterla decodificare. Eh, ma non va bene: in una stream non posso permettermi tutto ciò, e inoltre, se la codifica è veramente grande, potrei non riuscire a tenerla tutta in memoria.

Una prima soluzione sono i **codici a virgola**, ovvero codici che hanno un carattere di terminazione per dire quando una parola è finita, e quindi risolvere il problema stream negli UD, ma noi faremo altro.

Definizione 1.3 (*Codici istantanei*): Un codice c è istantaneo se ogni parola di codice non è prefissa di altre parole di codice.

I CI hanno la proprietà che stiamo cercando, ovvero permettono una decodifica stream, quindi non dobbiamo aspettare tutta la codifica prima di passare alla decodifica, ma possiamo farla appena riconosciamo una parola di codice. Inoltre, per verificare se un codice è CI devo solo controllare se vale la regola dei prefissi, e questa è molto più veloce rispetto al controllo che dovevo fare negli UD.

Ho quindi la seguente gerarchia:

$$\text{CI} \subset \text{UD} \subset \text{NS} \subset \text{codici} .$$

Ritorniamo all'obiettivo di prima: minimizzare la quantità

$$\mathbb{E}[l_c] = \sum_{x \in X} l_c(x) p(x).$$

Vediamo come, mettendo ogni volta dei nuovi vincoli sul codice, questo valore atteso aumenta, visto che vogliamo dei codici con buone proprietà ma questo va sprecato in termini di bit utilizzati.

Teorema 1.1: Se c è un CI allora c è un UD.

Dimostrazione 1.1: Dimostro il contrario, ovvero che se un codice non è UD allora non è CI.

Sia $c : X \longrightarrow D^+$ e sia C la sua estensione. Assumiamo che c sia non singolare. Se c non è UD allora esistono due messaggi x_1 e x_2 diversi che hanno la stessa codifica $C(x_1) = C(x_2)$.

Per mantenere i due messaggi diversi possono succede due cose:

- un messaggio è prefisso dell'altro: se x_1 è formato da x_2 e altri m caratteri, vuol dire che i restanti m caratteri di x_1 devono essere codificati con la parola vuota, che per definizione di codice non è possibile, e, soprattutto, la parola vuota sarebbe prefissa di ogni altra parola di codice, quindi il codice c non è istantaneo;
- esiste almeno una posizione in cui i due messaggi differiscono: sia i la prima posizione dove i due messaggi differiscono, ovvero $x_1[i] \neq x_2[i]$ e $x_1[j] = x_2[j]$ per $1 \leq j \leq i-1$, ma allora $c(x_1[i]) \neq c(x_2[i])$ e $c(x_1[j]) = c(x_2[j])$ perché c è non singolare, quindi per avere la stessa codifica devo tornare al punto 1 e avere x_1 come prefisso di x_2. Come visto poco fa, otteniamo che c non è istantaneo.

Ma allora c non è istantaneo. ■

2. Disuguaglianza di Kraft

Cosa possiamo dire sui CI? Sono molto graditi perché, oltre a identificare un CI in maniera molto semplice guardando i prefissi, possiamo capire **se esiste** un codice istantaneo senza vedere il codice, ovvero guardando solo le lunghezze delle parole di codice.

Questa è una differenza abissale: prima guardavamo le parole di codice e controllavo i prefissi, ora non ho le parole, posso guardare solo le lunghezze delle parole di codice e, in base a queste, posso dire se esiste un CI con quelle lunghezze. Il problema principale è che non so che codice è quello istantaneo con quelle lunghezze, sto solo dicendo che sono sicuro della sua esistenza.

Come mai seguire questa via e non quelle dei prefissi? Se il codice è molto grande, guardare questa proprietà è meno oneroso rispetto al guardare i prefissi.

La proprietà che stiamo blaterando da un po' è detta **disuguaglianza di Kraft**.

Teorema 2.1 (*Disuguaglianza di Kraft*): Dati una sorgente $X = \{x_1, ..., x_m\}$ di m simboli, $d > 1$ base del codice e m interi $l_1, ..., l_m > 0$ che mi rappresentano le lunghezze dei simboli del codice, esiste un CI

$$c : X \longrightarrow D^+$$

tale che

$$l_c(x_i) = l_i \quad \forall i = 1, ..., m$$

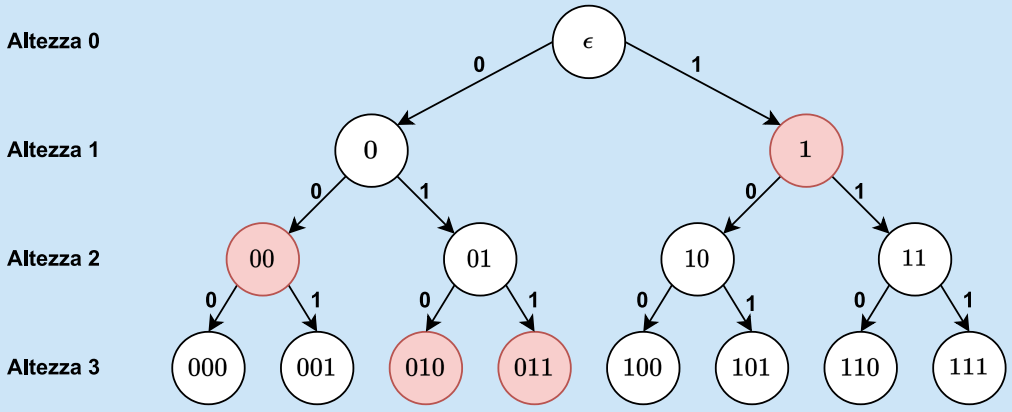
se e solo se

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

Dimostrazione 2.1: Dimostriamo la doppia implicazione.

($\Longrightarrow$) Esiste un CI con quelle proprietà, dimostro che vale la disuguaglianza di Kraft.

Sia c il nostro CI e d la base di c, dobbiamo definire la profondità del nostro codice. Possiamo usare un albero di codifica, ovvero un albero d-ario che rappresenta come sono codificate le varie parole di codice. Vediamo un breve esempio.



Vogliamo sapere

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

Costruiamo l'albero di codifica per il nostro CI e mettiamo dentro questo albero le parole del nostro codice. Come lo costruiamo? Creiamo l'albero d-ario alto $l_{\max}$ completo e scegliamo, in questo albero, delle parole ad altezza $l_i \quad \forall i = 1, \dots, m$.

Dividiamo il nostro albero in sotto-alberi grazie alle parole che abbiamo inserito: ogni sotto-albero ha come radice una delle parole che abbiamo scelto. Contiamo quante foglie sono coperte da ogni sotto-albero. Sono tutti alberi disgiunti, visto che non posso avere prefissi essendo c un CI. Il numero massimo di foglie è $d^{l_{\max}}$, ma noi potremmo non coprirle tutte, quindi

$$\sum_{i=1}^{m} |A_i| \leq d^{l_{\max}},$$

con A_i sotto-albero con radice la parola x_i. Notiamo che

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

Ora possiamo dividere tutto per $d^{l_{\max}}$ e ottenere

$$\sum_{i=1}^{m} \frac{d^{l_{\max}-l_i}}{d^{l_{\max}}} = \sum_{i=1}^{m} d^{-l_i} \leq 1.$$

($\Longleftarrow$) Vale la disuguaglianza di Kraft, dimostro che esiste un CI con quelle proprietà.

Abbiamo le lunghezze che rendono vera la disuguaglianza di Kraft, quindi costruiamo l'albero di codifica: scegliamo un nodo ad altezza l_i e poi proibiamo a tutti gli altri nodi ancora da inserire di:

- inserirsi nel sotto-albero di nodi già selezionati;
- scegliere nodi presenti nel cammino da un nodo già selezionato fino alla radice.

Una volta costruito l'albero vedo che esso rappresenta un CI, perché tutti i nodi non hanno nodi nel loro sotto-albero, quindi non ho prefissi per costruzione, quindi questo è un CI. ■

3. Entropia

3.1. Codice di Shannon-Fano

Il nostro obiettivo rimane quello di cercare il codice migliore, quello che minimizza il valore atteso delle lunghezze di tale codice, ovvero vogliamo trovare delle lunghezze $l_1, \dots, l_m$ tali che

$$\min_{l_1,\dots,l_m} \sum_{i=1}^{m} l_i p_i.$$

Non voglio solo minimizzare, ora abbiamo a disposizione anche la disuguaglianza di Kraft: la andiamo ad imporre, così da avere anche un CI. Devono valere contemporaneamente

$$\begin{cases} \min\limits_{l_1,\dots,l_m} \sum\limits_{i=1}^{m} l_i p_i \\ \sum\limits_{i=1}^{m} d^{-l_i} \leq 1 \end{cases}.$$

Notiamo che

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

Per comodità, associamo p_i al simbolo x_i con lunghezza l_i. Allora guardiamo tutti i singoli valori della sommatoria, perché *«se rispetto i singoli rispetto anche le somme»*, quindi

$$d^{-l_i} \leq p_i \quad \forall i = 1, \dots, m$$
$$l_i \geq \log_d\left(\frac{1}{p_i}\right).$$

Con questa relazione ho appena detto come sono le lunghezze l_i del mio codice: sono esattamente

$$l_i = \left\lceil \log_d\left(\frac{1}{p_i}\right) \right\rceil.$$

Posso quindi costruire un codice mettendo in relazione le lunghezze del codice con la probabilità di estrarle dalla sorgente. Il valore atteso ora diventa

$$\mathbb{E}[l_c] = \sum_{i=1}^{m} p_i \left\lceil \log_d\left(\frac{1}{p_i}\right) \right\rceil.$$

grazie alla nuova definizione delle lunghezze delle parole di codice. Notiamo come il valore atteso delle lunghezze dipenda solo dalla distribuzione di probabilità dei simboli sorgente.

La tecnica che associa ad ogni lunghezza un valore in base alla probabilità ci consente di generare un codice chiamato **codice di Shannon** (*o Shannon-Fano*).

Questa costruzione dei codici di Shannon-Fano è molto bella:

- se ho una probabilità *bassa* di estrarre il simbolo x_i allora la sua codifica è molto lunga;
- se ho una probabilità *alta* di estrarre il simbolo x_i allora la sua codifica è molto corta.

Purtroppo, i codici di Shannon-Fano **non sono ottimali**.

Esempio 3.1.1: Sia X una sorgente di due simboli x_1 e x_2 con probabilità

$$p_1 = 0.1 \quad | \quad p_2 = 0.9.$$

Il codice di Shannon risultante ha lunghezze

$$l_1 = \left\lceil \log_2\left(\frac{1}{0.1}\right) \right\rceil = 4 \quad | \quad l_2 = \left\lceil \log_2\left(\frac{1}{0.9}\right) \right\rceil = 1.$$

Questo sicuramente non è ottimale: basta dare 0 a x_0 e 1 a x_1 per avere un codice ottimo.

3.2. Entropia e tanti teoremi carini

La quantità

$$\sum_{i=1}^{m} p_i \log_d\left(\frac{1}{p_i}\right)$$

viene chiamata **entropia**. Come vedremo, sarà una misura che definisce quanto i nostri codici possono essere compressi, una sorta di misura di **compattezza**: oltre quella soglia non posso andare sennò perdo informazione.

Data la sorgente $\langle X, p\rangle$ associamo ad essa la variabile casuale

$$\mathbb{X} : X \longrightarrow \{a_1, ..., a_m\}$$

tale che $P(\mathbb{X} = a_1) = p_i$. Abbiamo l'entropia d-aria

$$H_d(\mathbb{X}) = \sum_{i=1}^{m} p_i \log_d\left(\frac{1}{p_i}\right)$$

che dipende solamente dalla distribuzione di probabilità della mia sorgente.

Vogliamo cambiare la base dell'entropia, come facciamo? Ricordiamo che

$$\log_d(p) = \frac{\ln(p)}{\ln(d)} = \frac{\ln(p)}{\ln(d)} \cdot \frac{\ln(a)}{\ln(a)} = \frac{\ln(p)}{\ln(a)} \cdot \frac{\ln(a)}{\ln(d)} = \log_a(p) \log_d(a)$$

quindi

$$H_d(\mathbb{X}) = \log_d(a) H_a(\mathbb{X}),$$

ovvero per cambiare la base dell'entropia pago una costante moltiplicativa $\log_d(a)$.

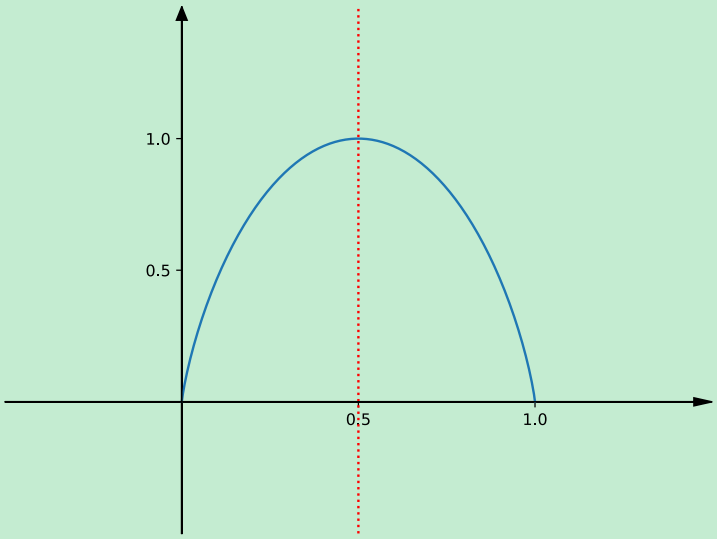
Esempio 3.2.1 (*Entropia binaria*): Sia $\mathbb{X}$ una variabile aleatoria bernoulliana tale che

$$P(\mathbb{X} = 1) = p \quad | \quad P(\mathbb{X} = 0) = 1 - p.$$

Calcoliamo l'entropia

$$H_2(\mathbb{X}) = p \log_2\left(\frac{1}{p}\right) + (1-p) \log_2\left(\frac{1}{1-p}\right)$$

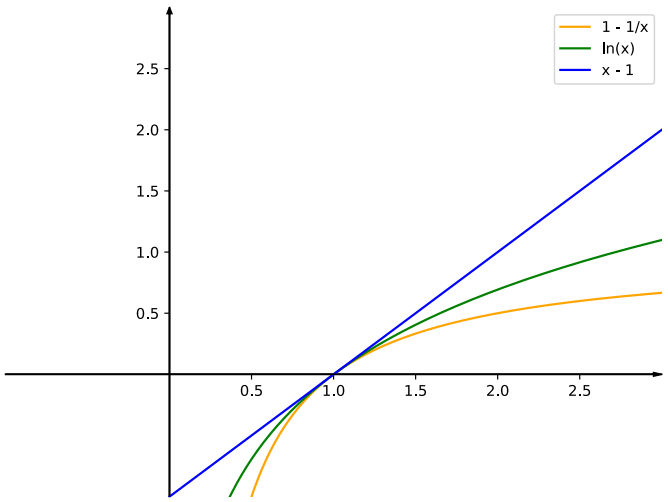
e vediamo quanto vale nell'intervallo $(0, 1)$.



Ce lo potevamo aspettare? **SI**: nel caso di evento certo e evento impossibile io so esattamente quello che mi aspetta, quindi l'informazione è nulla, mentre in caso di totale incertezza l'informazione che posso aspettarmi è massima.

Come mai possiamo affermare quanto detto nell'ultimo esempio? Quando ho una serie di eventi, se gli eventi mi danno **totale incertezza** io non saprò mai quello che potrebbe succedere, quindi l'informazione che mi dà l'osservazione dell'evento è massima. Al contrario, se ho eventi certi o impossibile l'informazione è nulla, visto che so già l'esito dell'osservazione.

Vediamo ora due bound del logaritmo che ci permetteranno di dimostrare tanti bei teoremi.



Come vediamo dal grafico abbiamo che

$$1 - \frac{1}{x} \leq \ln(x) \leq x - 1.$$

Teorema 3.2.1: Sia $\mathbb{X}$ una variabile casuale che assume i valori $\{a_1, ..., a_m\}$, allora

$$H_d(\mathbb{X}) \leq \log_d(m) \quad \forall d > 1.$$

In particolare,

$$H_d(\mathbb{X}) = \log_d(m)$$

se e solo se $\mathbb{X}$ è distribuita secondo un modello uniforme su $\{a_1, ..., a_m\}$.

Dimostrazione 3.2.1: Andiamo a valutare $H_d(\mathbb{X}) - \log_d(m)$ per vedere che valore assume.

Consideriamo base dell'entropia e d come lo stesso valore: se così non fosse avremmo un fattore moltiplicativo davanti all'entropia che però non cambia la dimostrazione successiva.

Valutiamo quindi

$$\begin{aligned} H_d(\mathbb{X}) - \log_d(m) &= \sum_{i=1}^{m} p_i \log_d\left(\frac{1}{p_i}\right) - \log_d(m) = \\ &= \sum_{i=1}^{m} p_i \log_d\left(\frac{1}{p_i}\right) - \underbrace{\sum_{i=1}^{m} p_i}_{=1} \log_d(m) = \\ &= \sum_{i=1}^{m} p_i \left(\log_d\left(\frac{1}{p_i}\right) - \log_d(m)\right) = \\ &= \sum_{i=1}^{m} p_i \log_d\left(\frac{1}{p_i m}\right). \end{aligned}$$

Sappiamo che $\ln(n) \leq x - 1$, quindi

$$H_d(\mathbb{X}) - \log_d(m) \leq \sum_{i=1}^{m} p_i \left(\frac{1}{p_i m} - 1\right) = \sum_{i=1}^{m} \frac{1}{m} - \sum_{i=1}^{m} p_i = 1 - 1 = 0.$$

Ma allora

$$H_d(\mathbb{X}) - \log_d(m) \leq 0 \Longrightarrow H_d(\mathbb{X}) \leq \log_d(m).$$

In particolare, se

$$P(X = a_i) = \frac{1}{m} \quad \forall i = 1, ..., m$$

allora

$$H_d(\mathbb{X}) = \sum_{i=1}^{m} \frac{1}{m} \log_d(m) = m\frac{1}{m} \log_d(m) = \log_d(m).$$

■

Abbiamo detto che l'entropia ci dice quanto possiamo compattare il nostro messaggio prima che iniziamo a perdere informazioni: per dimostrare questo bound introduciamo prima la nozione di entropia relativa.

Siano $\mathbb{X}, \mathbb{W}$ due variabili aleatorie definite sullo stesso dominio S, e siano $p_{\mathbb{X}}$ e $p_{\mathbb{W}}$ le distribuzioni di probabilita delle due variabili aleatorie, allora l'**entropia relativa**

$$\mathbb{D}(\mathbb{X} \parallel \mathbb{W}) = \sum_{s \in S} p_{\mathbb{X}}(s) \log_d \left(\frac{p_{\mathbb{X}}(s)}{p_{\mathbb{W}}(s)} \right)$$

è la quantità che misura la distanza che esiste tra le variabili aleatorie $\mathbb{X}$ e $\mathbb{W}$. In generale vale

$$\mathbb{D}(\mathbb{X} \parallel \mathbb{W}) \neq \mathbb{D}(\mathbb{W} \parallel \mathbb{X})$$

perché essa non è una distanza metrica, ma solo una distanza in termini di diversità.

Teorema 3.2.2 (*Information inequality*):

$$\mathbb{D}(\mathbb{X} \parallel \mathbb{W}) \geq 0.$$

Dimostrazione 3.2.2: Come nella dimostrazione precedente, se volessimo cambiare la base del logaritmo avremmo un fattore moltiplicativo davanti alla sommatoria, che però non cambia la dimostrazione. Noi manteniamo la base d in questa dimostrazione.

Valutiamo, sapendo che $1 - \frac{1}{x} \leq \ln(x)$, la quantità

$$\begin{aligned} \sum_{s \in S} p_{\mathbb{X}}(s) \log_d \left(\frac{p_{\mathbb{X}}(s)}{p_{\mathbb{W}}(s)} \right) &\geq \sum_{s \in S} p_{\mathbb{X}}(s) \left(1 - \frac{p_{\mathbb{W}}(s)}{p_{\mathbb{X}}(s)} \right) = \\ &= \sum_{s \in S} p_{\mathbb{X}}(s) - p_{\mathbb{W}}(s) = \\ &= \sum_{s \in S} p_{\mathbb{X}}(s) - \sum_{s \in S} p_{\mathbb{W}}(s) = 1 - 1 = 0. \end{aligned}$$

Ma allora

$$\mathbb{D}(\mathbb{X} \parallel \mathbb{W}) \geq 0.$$ ■

Se $\mathbb{D}(\mathbb{X} \parallel \mathbb{W}) = 0$ allora ho spakkato, vuol dire che non ho distanza tra le due variabili aleatorie.

Vediamo finalmente il bound che ci dà l'entropia sulla compattezza del codice.

Teorema 3.2.3: Sia $c : X \longrightarrow D^+$ un CI d-ario per la sorgente $\langle X, p \rangle$, allora

$$\mathbb{E}[l_c] \geq H_d(\mathbb{X}).$$

Dimostrazione 3.2.3: Chiamo $\mathbb{W} : X \longrightarrow \mathbb{R}$ una variabile casuale con una distribuzione di probabilita $q(x)$ tale che

$$q(x) = \frac{d^{-l_c(x)}}{\sum_{x' \in X} d^{-l_c(x')}}.$$

Valutiamo

$$\begin{aligned}
\mathbb{E}[l_c] - H_d(\mathbb{X}) &= \sum_{x \in X} l_c(x)p(x) - \sum_{x \in X} p(x) \log_d \left(\frac{1}{p(x)} \right) = \\
&= \sum_{x \in X} p(x) \left(l_c(x) - \log_d \left(\frac{1}{p(x)} \right) \right) = \\
&= \sum_{x \in X} p(x) \left(\log_d d^{l_c(x)} - \log_d \left(\frac{1}{p(x)} \right) \right) = \\
&= \sum_{x \in X} p(x) \log_d \left(p(x) d^{l_c(x)} \right) = \\
&= \textit{massaggiamo pesantemente la formula} = \\
&= \sum_{x \in X} p(x) \log_d \left(\frac{p(x)}{d^{-l_c(x)}} \right) = \\
&= \sum_{x \in X} p(x) \log_d \left(\frac{p(x)}{d^{-l_c(x)}} \cdot \frac{\sum_{x' \in X} d^{-l_c(x')}}{\sum_{x' \in X} d^{-l_c(x')}} \right) = \\
&= \sum_{x \in X} p(x) \log_d \left(\frac{p(x)}{q(x)} \right) + \sum_{x \in X} p(x) \log_d \left(\frac{1}{\sum_{x' \in X} d^{-l_c(x')}} \right) = \\
&= \mathbb{D}(\mathbb{X} \parallel \mathbb{W}) + \log_d \left(\frac{1}{\sum_{x' \in X} d^{-l_c(x')}} \right) \sum_{x \in X} p(x) = \\
&= \mathbb{D}(\mathbb{X} \parallel \mathbb{W}) + \log_d \left(\frac{1}{\sum_{x' \in X} d^{-l_c(x')}} \right).
\end{aligned}$$

Per l'information inequality sappiamo che

$$\mathbb{D}(\mathbb{X} \parallel \mathbb{W}) \geq 0,$$

mentre per la disuguaglianza di Kraft sappiamo che

$$\sum_{x \in X} d^{-l_c(x)} \leq 1$$

e quindi possiamo dire che

$$\frac{1}{\sum_{x \in X} d^{-l_c(x)}} \geq 1 \Longrightarrow \log_d (\geq 1) \geq 0.$$

Otteniamo quindi che

$$\mathbb{E}[l_c] - H_d(\mathbb{X}) \geq 0 \Longrightarrow \mathbb{E}[l_c] \geq H_d(\mathbb{X}).$$ ■

Questo bound ci dice che un codice non può comunicare meno di quanto vale l'entropia di quella sorgente, indipendentemente dal codice scelto.

Abbiamo quasi finito: vediamo una proprietà del codice di Shannon, ora che abbiamo conosciuto abbastanza bene il concetto di entropia e tutti i suoi bound.

Teorema 3.2.4: Per ogni sorgente $\langle X, p\rangle$ con:
- $X = \{x_1, ..., x_m\}$ insieme dei simboli sorgente;
- $P = \{p_1, ..., p_m\}$ probabilità associate ai simboli di X;
- $c : X \longrightarrow D^+$ codice di Shannon con lunghezze $l_1, ..., l_m$ tali che $l_i = l_c(x_i)$ costruite con

$$l_i = \left\lceil \log_d\left(\frac{1}{p_i}\right)\right\rceil \quad \forall i = 1, ..., m$$

. Vale la relazione

$$\mathbb{E}[l_c] < H_d(\mathbb{X}) + 1.$$

Dimostrazione 3.2.4: Verifichiamo che

$$\begin{aligned}\mathbb{E}[l_c] &= \sum_{i=1}^{m} p_i l_i = \sum_{i=1}^{m} p_i \left\lceil \log_d\left(\frac{1}{p_i}\right)\right\rceil \\ &< \sum_{i=1}^{m} p_i \left(\log_d\left(\frac{1}{p_i}\right) + 1\right) = \sum_{i=1}^{m} p_i \log_d\left(\frac{1}{p_i}\right) + \sum_{i=1}^{m} p_i = H_d(\mathbb{X}) + 1.\end{aligned}$$

Che fastidio questo quadrato. ■

Ho trovato quindi un bound superiore alle lunghezze delle parole di codice di un codice di Shannon. Se uniamo questo bound a quelli di questo capitolo otteniamo

$$H_d(\mathbb{X}) \leq \mathbb{E}[l_c] \leq H_d(\mathbb{X}) + 1.$$

Per finire, cosa succede se per la sorgente $\langle X, p\rangle$ non ho la distribuzione p ma ho una distribuzione q di una variabile casuale $\mathbb{Y}$ che ha campionato $\mathbb{X}$?

Teorema 3.2.5: Data una sorgente $\langle X, p\rangle$, se $c : X \longrightarrow D^+$ è un codice di Shannon avente lunghezze

$$l_c(x) = \left\lceil \log_d \frac{1}{q(x)}\right\rceil$$

dove q è una distribuzione di probabilità su X campionata con $\mathbb{Y}$, allora

$$\mathbb{E}[l_c] < H_d(\mathbb{X}) + 1 + \mathbb{D}(\mathbb{X} \parallel \mathbb{Y}).$$

Dimostrazione 3.2.5: Dimostrazione abbastanza banale:

$$\begin{aligned}
\mathbb{E}[l_c] &= \sum_{x\in X} p(x)\left\lceil \log_d\left(\frac{1}{q(x)}\right)\right\rceil \\
&< \sum_{x\in X} p(x)\left(\log_d\left(\frac{1}{q(x)}\right)+1\right) = \\
&= \sum_{x\in X} p(x)\log_d\left(\frac{1}{q(x)}\right) + \sum_{x\in X} p(x) = \\
&= \sum_{x\in X} p(x)\log_d\left(\frac{p(x)}{p(x)q(x)}\right) + 1 = \\
&= \sum_{x\in X} p(x)\log_d\left(\frac{p(x)}{q(x)}\right) + \sum_{x\in X} p(x)\log_d\left(\frac{1}{p(x)}\right) + 1 = \\
&= H_d(\mathbb{X}) + 1 + \mathbb{D}(\mathbb{X} \parallel \mathbb{Y}).
\end{aligned}$$

Odio il quadratino. ■

4. Primo teorema di Shannon

Nello scorso capitolo abbiamo visto come il codice di Shannon obbedisca ai seguenti bound:

$$H_d(\mathbb{X}) \leq \mathbb{E}[l_c] \leq H_d(\mathbb{X}) + 1.$$

Iniziamo a vedere però qualche problematica: questa risiede nel $+1$ dell'upper bound. Ogni volta che facciamo la codifica di un singolo carattere perdiamo poco, e questo *«poco»* è dato da quel $+1$. È una perdita locale, ma quando consideriamo un'intera parola o un intero messaggio la perdita conta.

Shannon si accorge di questa problematica perché usando l'estensione $C : X^+ \longrightarrow D^+$ la lunghezza delle parole di codice è data da

$$l_C(x_1 \cdot \ldots \cdot x_n) = \sum_{i=1}^{n} \left\lceil \log_d \left(\frac{1}{p_i} \right) \right\rceil.$$

Shannon allora propone una soluzione, che come vedremo non funzionerà nel caso reale. Vediamo un esempio per capire tutto ciò.

Esempio 4.1: Siano:

- $l_c(x_1) = \lceil 2.5 \rceil \longrightarrow 3$;
- $l_c(x_2) = \lceil 2.1 \rceil \longrightarrow 3$;
- $l_c(x_3) = \lceil 2.1 \rceil \longrightarrow 3$.

Come lunghezza totale della parola $x = x_1 x_2 x_3$ abbiamo 9.

Se pago al più un bit su ogni oggetto, il trucco che posso fare è fare il $\lceil\ \rceil$ della somma delle lunghezze, non la somma dei $\lceil\ \rceil$ delle lunghezze. In questo caso otteniamo $\lceil 2.5 + 2.1 + 2.1 \rceil = \lceil 6.7 \rceil = 7$, che è minore di 9. In poche parole ho *«spostato»* la sommatoria dentro $\lceil\ \rceil$.

Creiamo un nuovo codice: sia C_n un **codice a blocchi**, ed è un codice tale che

$$C_n : X^n \longrightarrow D^+.$$

Sembra una soluzione fantastica, ma cosa ci nasconde Shannon? Vediamolo con un altro esempio.

Esempio 4.2: Sia $X = \{0, \ldots, 9\}$, ho i codici c e C e mi viene chiesto di scrivere C_n con $d = 2$ e $n = 5$. Dove risiede il problema?

Dovrei codificare un numero enorme di parole di codice, in questo caso abbiamo 10^5.

Ecco cosa stava nascondendo Shannon: la **complessità** del codice è enorme.

Prendiamo un messaggio e lo dividiamo in blocchi di dimensione n numerandoli da 1 a m. Questi blocchi sono estratti dalla nostra sorgente $\langle X, p \rangle$. Ogni blocco è nella forma $(x_1, \ldots, x_n)$ e contiene n simboli estratti tramite estrazioni indipendenti e identicamente distribuite, ovvero

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p_i.$$

Questa quantità verrà indicata con

$$P_n(x_1, \ldots, x_n).$$

Definisco una nuova sorgente $\langle X^n, P_n \rangle$ sulla quale definisco il mio codice a blocchi C_n. È una sorgente che pesca n oggetti da $\langle X, p \rangle$. Vediamo cosa succede all'entropia di questa nuova sorgente.

Ho n variabili casuali (*mani*), ognuna che estrae un simbolo da X: devo calcolare quindi

$$\begin{aligned} H_d(\mathbb{X}_1, ..., \mathbb{X}_n) &= \sum_{x_1,...,x_n} P_n(x_1, ..., x_n) \log_d\left(\frac{1}{P_n(x_1, ..., x_n)}\right) = \\ &= \sum_{x_1} \cdots \sum_{x_n} \left(\prod_{i=1}^{n} p(x_i)\right)\left(\log_d\left(\frac{1}{\prod_{i=1}^{n} p(x_i)}\right)\right) = \\ &= \sum_{x_1} \cdots \sum_{x_n} \left(\prod_{i=1}^{n} p(x_i)\right)\left(\log_d\left(\prod_{i=1}^{n} p(x_i)^{-1}\right)\right) = \\ &= \sum_{x_1} \cdots \sum_{x_n} \left(\prod_{i=1}^{n} p(x_i)\right)\left(\sum_{i=1}^{n} \log_d\left(\frac{1}{p(x_i)}\right)\right). \end{aligned}$$

Come semplificare questo schifo? Vediamo il caso generale con un esempio.

Esempio 4.3: Sia $n = 2$, andiamo a calcolare l'entropia come

$$\begin{aligned} H_d(\mathbb{X}_1, \mathbb{X}_2) &= \sum_{x_1}\sum_{x_2} \left(\prod_{i=1}^{2} p(x_i)\right)\left(\sum_{i=1}^{2} \log_d\left(\frac{1}{p(x_i)}\right)\right) = \\ &= \sum_{x_1}\sum_{x_2} (p(x_1)p(x_2))\left(\log_d\left(\frac{1}{p(x_1)}\right) + \log_d\left(\frac{1}{p(x_2)}\right)\right) = \\ &= \sum_{x_1}\sum_{x_2} (p(x_1)p(x_2)) \log_d\left(\frac{1}{p(x_1)}\right) + p(x_1)p(x_2) \log_d\left(\frac{1}{p(x_2)}\right) = \\ &= \sum_{x_1}\sum_{x_2} (p(x_1)p(x_2)) \log_d\left(\frac{1}{p(x_1)}\right) + \sum_{x_1}\sum_{x_2} p(x_1)p(x_2) \log_d\left(\frac{1}{p(x_2)}\right) = \\ &= \left(\sum_{x_1} p(x_1) \log_d\left(\frac{1}{p(x_1)}\right)\right)\left(\sum_{x_2} p(x_2)\right) + \\ &+ \left(\sum_{x_1} p(x_1)\right)\left(\sum_{x_2} p(x_2) \log_d\left(\frac{1}{p(x_2)}\right)\right) = \\ &= H_d(\mathbb{X}_1) \cdot 1 + 1 \cdot H_d(\mathbb{X}_2) = H_d(\mathbb{X}_1) + H_d(\mathbb{X}_2). \end{aligned}$$

Ma allora

$$\begin{aligned} H_d(\mathbb{X}_1, ..., \mathbb{X}_n) &= \sum_{x_1} \cdots \sum_{x_n} \left(\prod_{i=1}^{n} p(x_i)\right)\left(\sum_{i=1}^{n} \log_d\left(\frac{1}{p(x_i)}\right)\right) = \\ &= H_d(\mathbb{X}_1) + ... + H_d(\mathbb{X}_n) = nH_d(\mathbb{X}). \end{aligned}$$

L'ultimo passaggio è vero perché tutte le variabili casuali stanno pescando dalla stessa sorgente con le stesse probabilità.

Teorema 4.1 (*Primo teorema di Shannon*): Sia $C_n : X^n \longrightarrow D^+$ codice a blocchi di Shannon d-ario per la sorgente $\langle X, p \rangle$ con

$$l_{C_n}(x_1, ..., x_n) = \left\lceil \log_d \frac{1}{P_n(x_1, ..., x_n)} \right\rceil,$$

allora

$$\lim_{n\to\infty} \frac{1}{n} \mathbb{E}\left[l_{C_n}\right] = H_d(\mathbb{X}).$$

Dimostrazione 4.1: La dimostrazione è banale:

$$\begin{aligned}
H_d(\mathbb{X}_1, ..., \mathbb{X}_n) \leq \mathbb{E}\left[l_{C_n}\right] &< H_d(\mathbb{X}_1, ..., \mathbb{X}_n) + 1 \\
nH_d(\mathbb{X}) \leq \mathbb{E}\left[l_{C_n}\right] &\leq nH_d(\mathbb{X}) + 1 \\
\frac{1}{n}(nH_d(\mathbb{X})) \leq \frac{1}{n}\mathbb{E}\left[l_{C_n}\right] &\leq \frac{1}{n}(nH_d(\mathbb{X}) + 1) \\
H_d(\mathbb{X}) \leq \frac{1}{n}\mathbb{E}\left[l_{C_n}\right] &\leq H_d(\mathbb{X}) + \frac{1}{n}.
\end{aligned}$$

Se passo al limite per $n \to \infty$, per il teorema dei due carabinieri vale

$$\lim_{n\to\infty} \frac{1}{n} \mathbb{E}\left[l_{C_n}\right] = H_d(\mathbb{X}).$$ ■

Questo teorema ci dice che se tendiamo a ∞ la grandezza del blocco n allora il codice è ottimale, perché è uguale al lower bound che avevamo definito per $\mathbb{E}\left[l_{C_n}\right]$.

5. Codice di Huffman

Il codice di Shannon che abbiamo visto nel capitolo dell'entropia era un codice molto interessante, aveva dei buoni bound ma non era il CI ottimale. Inoltre, grazie al primo teorema di Shannon, questo codice risultava impraticabile se la grandezza del blocco cresceva.

Vediamo il **codice di Huffman**. Data una sorgente $\langle X, p \rangle$ e dato $d > 1$, l'algoritmo per creare il codice di Huffman per la sorgente X esegue i seguenti passi:

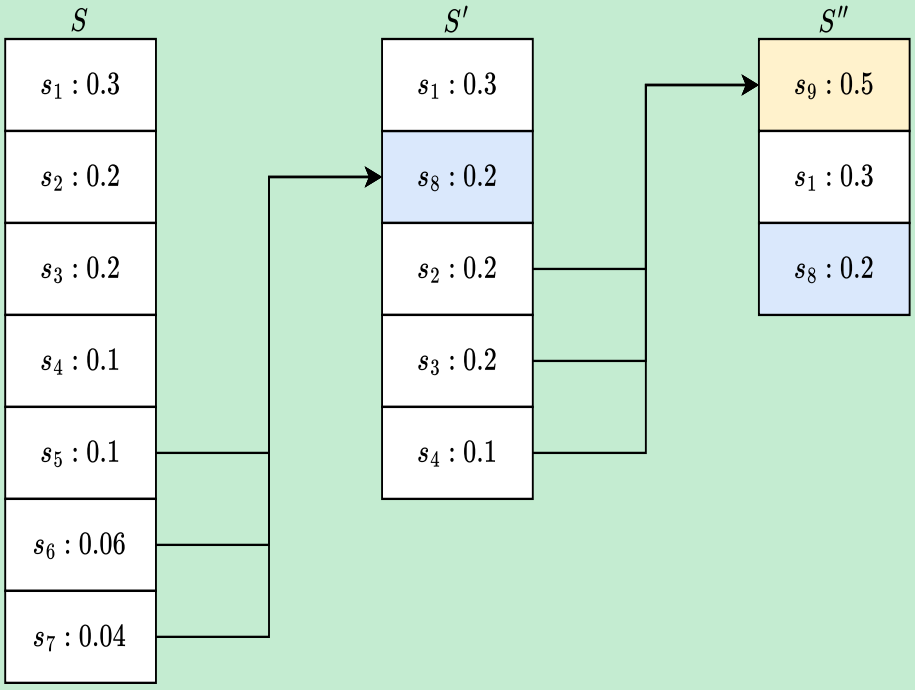
1. ordina le probabilità p_i in maniera decrescente;
2. rimuovi i d simboli meno probabili da X e crea una nuova sorgente aggiungendo un nuovo simbolo che abbia come probabilità la somma delle probabilità dei simboli rimossi;
3. se la nuova sorgente ha più di d simboli torna al punto 1.

Esempio 5.1: Data la sorgente $\langle S, p \rangle$ con:

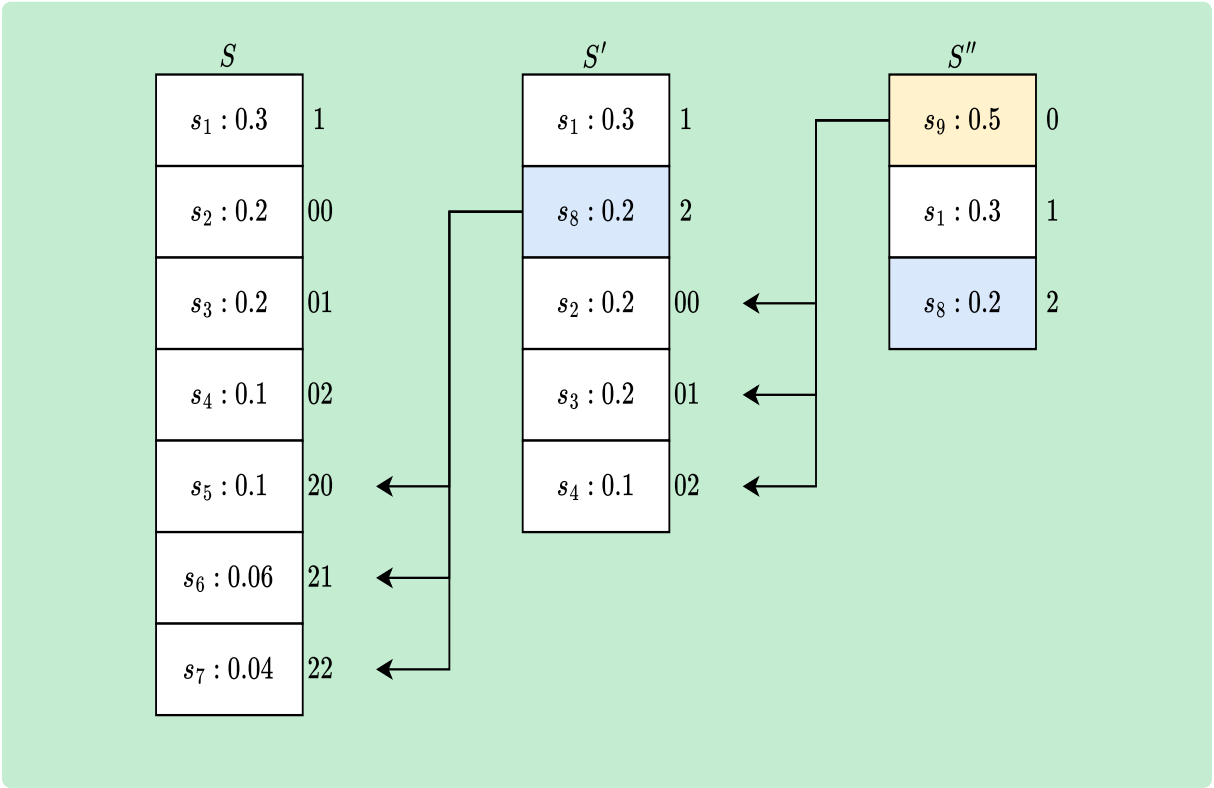
- $S = \{s_1, ..., s_7\}$ e
- $P = \{0.3, 0.2, 0.2, 0.1, 0.1, 0.06, 0.04\}$,

trovare il codice di Huffman ternario per questa sorgente.

Nella prima fase andiamo a comprimere i simboli meno probabili arrivando a definire la sorgente S'' dopo due iterazioni dell'algoritmo.



Nella seconda fase invece andiamo a fare un rollback: a partire dall'ultima sorgente creata andiamo ad assegnare i simboli di D ai simboli sorgente correnti, propaghiamo i simboli alla sorgente precedente e andiamo ad eseguire lo stesso procedimento per i d simboli che sono stati compattati.



Questo codice è un codice che a noi piace molto perché rispetta due condizioni per noi fondamentali: minimizza il valore atteso delle lunghezze delle parole di codice e rispetta Kraft. Infatti:

$$\text{CH} = \begin{cases} \min\limits_{l_1,\dots,l_m} \sum\limits_{i=1}^{m} l_i p_i \\ \sum\limits_{i=1}^{m} d^{-l_i} \leq 1 \end{cases}.$$

Lemma 5.1: Sia c un codice d-ario di Huffman per la sorgente $\langle X', p' \rangle$ con $X' = \{x_1, \dots, x_{m-d+1}\}$ e probabilità $p_1 \geq \dots \geq p_{m-d+1}$. Data ora la sorgente $\langle X, p \rangle$ con $X = \{x_1, \dots, x_m\}$ costruita da X' togliendo il simbolo x_k e aggiungendo d simboli $x_{m-d+2}, \dots, x_m$ con probabilità $p_k \geq p_{m-d+2} \geq \dots \geq p_m$ tali che

$$\sum_{i=2}^{d} p_{m-d+i} = p_k.$$

Allora il codice

$$c(x_t) = \begin{cases} c'(x_t) & \text{se } t \neq k \\ c'(x_k) \cdot i & \text{se } t \in \{m-d+2, \dots, m\} \wedge \forall i \in D \end{cases}$$

è un codice di Huffman per la sorgente X.

Grazie a questo lemma ora possiamo dimostrare un risultato importante sul codice di Huffman.

Teorema 5.1: Data la sorgente $\langle X, p \rangle$ e dato $d > 1$, il codice d-ario di Huffman c minimizza $\mathbb{E}[l_c]$ tra tutti i codici d-ari per la medesima sorgente.

Dimostrazione 5.1: Dimostriamo per induzione su m.

Il passo base è $m = 2$, quindi abbiamo due simboli x_1 e x_2 che, indipendentemente dalla probabilità assegnatagli, avranno 0 e 1 come codifica, che è minima.

Assumiamo ora che Huffman sia ottimo per sorgenti di grandezza $m - 1$ e dimostriamo che sia ottimo per sorgenti di grandezza m.

Fissata $\langle X, p \rangle$ sorgente di m simboli, siano $u, v \in X$ i due simboli con le probabilità minime. Costruiamo la sorgente $\langle X', p' \rangle$ dove u, v sono sostituiti da z tale che

$$p'(x) = \begin{cases} p(x) & \text{se } x \neq z \\ p(u) + p(v) & \text{se } x = z \end{cases}.$$

Ho $m - 1$ simboli, quindi per ipotesi induttiva c' è un codice di Huffman ottimo. Per il lemma precedente, anche il codice c è di Huffman ed è tale che

$$c(x) = \begin{cases} c'(x) & \text{se } x \neq u \wedge x \neq v \\ c'(u) \cdot 0 & \text{se } x = u \\ c'(v) \cdot 1 & \text{se } x = v \end{cases}.$$

Dimostriamo che il codice c è ottimo.

Calcoliamo il valore atteso delle lunghezze delle parole del codice c come

$$\begin{aligned}
\mathbb{E}[l_c] &= \sum_{x \in X} l_c(x) p(x) = \\
&= \sum_{x \in X'} l_{c'}(x) p'(x) - l_{c'}(z) p'(z) + l_c(u) p(u) + l_c(v) p(v) = \\
&= \mathbb{E}[l_{c'}] - l_{c'}(z) p'(z) + (l_{c'}(z) + 1) p(u) + (l_{c'}(z) + 1) p(v) = \\
&= \mathbb{E}[l_{c'}] - l_{c'}(z) p'(z) + (l_{c'}(z) + 1)(p(u) + p(v)) = \\
&= \mathbb{E}[l_{c'}] - l_{c'}(z) p'(z) + (l_{c'}(z) + 1) p'(z) = \\
&= \mathbb{E}[l_{c'}] - l_{c'}(z) p(z) + l_{c'}(z) p'(z) + p'(z) = \\
&= \mathbb{E}[l_{c'}] + p'(z).
\end{aligned}$$

Per dimostrare l'ottimalità di c consideriamo un altro codice c_2 per la sorgente $\langle X, p \rangle$ e verifichiamo che $\mathbb{E}[l_c] \leq \mathbb{E}\left[l_{c_2}\right]$. Sia c_2 istantaneo per $\langle X, p \rangle$ e siano $r, s \in X$ tali che $l_{c_2}(r)$ e $l_{c_2}(s)$ sono massime. Senza perdita di generalità assumiamo che r, s siano fratelli nell'albero di codifica di c_2. Infatti:

- se sono fratelli GG, godo;
- se non sono fratelli ma uno tra r e s ha un fratello (*sia f fratello di s ad esempio*) andiamo a scegliere s e f al posto di s e r;

- se non sono fratelli perché sono su due livelli diversi (*a distanza uno*) possiamo sostituire la codifica di quello più basso con quella del padre e ritornare in una delle due situazioni precedenti.

Definiamo il codice $\overline{c}_2$ tale che

$$\overline{c}_2 = \begin{cases} c_2(x) & \text{se } x \notin \{u, v, r, s\} \\ c_2(u) & \text{se } x = r \\ c_2(r) & \text{se } x = u \\ c_2(v) & \text{se } x = s \\ c_2(s) & \text{se } x = v \end{cases}.$$

In poche parole, abbiamo scambiato r con u e s con v.

Analizziamo $\mathbb{E}\left[l_{\overline{c}_2}\right] - \mathbb{E}\left[l_{c_2}\right]$ per dimostrare che il primo è minore o uguale del secondo. Notiamo prima di tutto che i simboli $x \notin \{u, v, r, s\}$ non compaiono nel conto successivo: questo perché nei due codici hanno lo stesso contributo in probabilità e lunghezza, quindi consideriamo solo i simboli appena citati visto che vengono scambiati.

$$\begin{aligned} \mathbb{E}\left[l_{\overline{c}_2}\right] - \mathbb{E}\left[l_{c_2}\right] &= p(r)l_{c_2}(u) + p(u)l_{c_2}(r) + p(s)l_{c_2}(v) + p(v)l_{c_2}(s) \\ &\quad -p(r)l_{c_2}(r) - p(u)l_{c_2}(u) - p(s)l_{c_2}(s) - p(v)l_{c_2}(v) = \\ &= p(r)\Big(l_{c_2}(u) - l_{c_2}(r)\Big) + p(u)\Big(l_{c_2}(r) - l_{c_2}(u)\Big) + \\ &\quad +p(s)\Big(l_{c_2}(v) - l_{c_2}(s)\Big) + p(v)\Big(l_{c_2}(s) - l_{c_2}(v)\Big) = \\ &= (p(r) - p(u))\Big(l_{c_2}(u) - l_{c_2}(r)\Big) + (p(s) - p(v))\Big(l_{c_2}(v) - l_{c_2}(s)\Big). \end{aligned}$$

Ma noi sappiamo che:

- $p(r) - p(u) \geq 0$ dato che u è un simbolo a probabilità minima;
- $l_{c_2}(u) - l_{c_2}(r) \leq 0$ dato che r è un simbolo a lunghezza massima;
- $p(s) - p(v) \geq 0$ dato che v è un simbolo a probabilità minima;
- $l_{c_2}(v) - l_{c_2}(s) \leq 0$ dato che s è un simbolo a lunghezza massima.

Stiamo sommando due quantità negative, quindi

$$\mathbb{E}\left[l_{\overline{c}_2}\right] - \mathbb{E}\left[l_{c_2}\right] \leq 0 \Longrightarrow \mathbb{E}\left[l_{\overline{c}_2}\right] \leq \mathbb{E}\left[l_{c_2}\right].$$

Introduciamo ora il codice c_2' fatto come segue:

$$c_2' = \begin{cases} \overline{c}_2(x) & \text{se } x \neq z \\ \omega & \text{se } x = z \end{cases}.$$

Questo codice è definito sulla sorgente $\langle X', p' \rangle$. Ma allora

$$\begin{aligned}\mathbb{E}\left[l_{\bar{c}_2}\right] &= \sum_{x\in X' \mid x\neq z} p'(x)l_{\bar{c}_2}(x) + p(u)\left(l_{c'_2}(z)+1\right) + p(v)\left(l_{c'_2}(z)+1\right) = \\ &= \sum_{x\in X' \mid x\neq z} p'(x)l_{\bar{c}_2}(x) + p'(z)l_{c'_2}(z) + p'(z) = \\ &= \mathbb{E}\left[l_{c'_2}\right] + p'(z) \\ &\geq \mathbb{E}[l_{c'}] + p'(z).\end{aligned}$$

Mettendo insiemi i due risultati otteniamo

$$\mathbb{E}[l_c] = \mathbb{E}[l_{c'}] + p'(z) \leq \mathbb{E}\left[l_{c'_2}\right] + p'(z) = \mathbb{E}\left[l_{\bar{c}_2}\right] \leq \mathbb{E}\left[l_{c_2}\right].$$

Ma allora

$$\mathbb{E}[l_c] \leq \mathbb{E}\left[l_{c_2}\right].$$ ■

Bisogna fare un piccolo appunto sull'algoritmo che genera il codice di Huffman: esso genera il codice ottimo se e solo se il numero di simboli è corretto. Quale è il numero corretto? Mo ci arrivo, calma.

Supponiamo di partire da m simboli, ad ogni iterazione rimuoviamo $d-1$ simboli:

$$m \longrightarrow m-(d-1) \longrightarrow m-2(d-1) \longrightarrow \ldots \longrightarrow m-t(d-1).$$

Chiamiamo ■ $= m-t(d-1)$. Siamo arrivati ad avere ■ elementi, con ■ $\leq d$.

Noi vorremmo avere esattamente d elementi per avere un albero ben bilanciato e senza perdita di rami, quindi aggiungiamo a ■ un numero k di **simboli dummy** tali per cui ■ $+k = d$. I simboli dummy sono dei simboli particolari che hanno probabilità nulla e che usiamo solo come riempimento. Supponiamo di eseguire ancora un passo dell'algoritmo, quindi da ■ $+k$ andiamo a togliere $d-1$ elementi, lasciando la sorgente con un solo elemento.

Cosa abbiamo ottenuto? Ricordando che ■ $= m-t(d-1)$, abbiamo fatto vedere che:

$$\begin{aligned}\blacksquare + k - (d-1) &= 1 \\ m - t(d-1) + k - (d-1) &= 1 \\ m + k - (t+1)(d-1) &= 1.\end{aligned}$$

In poche parole, il numero m di simboli sorgente, aggiunto al numero k di simboli *«fantoccio»*, è congruo ad 1 modulo $(d-1)$, ovvero

$$m + k \equiv 1 \mod d-1.$$

6. Disuguaglianza di Kraft-McMillan

Per ora abbiamo sempre considerato dei CI perché ci davano una serie di proprietà belle, ma ce l'abbiamo un codice UD che abbia lunghezze minime e che magari sia meglio di un CI? Se rilasso il vincolo *«no-stream»* esiste negli UD una cosa buona come Huffman o anche di meglio?

Modifichiamo leggermente l'estensione C di un codice: sia ora

$$C_k : X^k \longrightarrow D^+$$

l'**estensione k-esima** del codice c, con $k \geq 1$, tale che

$$C_k(x_1 \cdot ... \cdot x_k) = c(x_1) \cdot ... \cdot c(x_k).$$

Le lunghezze di C_k sono

$$l_{C_k}(x_1 \cdot ... \cdot x_k) = \sum_{i=1}^{k} l_c(x_i).$$

Per convenzione sia

$$l_{\max} = \max_{i=1,...,k} l_c(x_i).$$

Ma allora

$$l_{C_k}(x_1, ..., x_k) \leq k l_{\max}.$$

Teorema 6.1 (*Disuguaglianza di Kraft-McMillan*): Data una sorgente $X = \{x_1, ..., x_m\}$, data $d > 1$ base del codice e dati m interi $l_1, ..., l_m > 0$ che mi rappresentano le lunghezze dei simboli del codice, esiste un codice UD

$$c : X \longrightarrow D^+$$

tale che

$$l_c(x_i) = l_i \quad \forall i = 1, ..., m$$

se e solo se

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

Uguale alla disuguaglianza di Kraft nei CI, ma qua viene definita negli UD.

Dimostrazione 6.1: Dimostriamo la doppia implicazione.

$\{\Longleftarrow\}$ Pezzo gratis: per la disuguaglianza di Kraft sui CI esiste un CI, ma i CI sono anche UD, quindi questa implicazione è vera.

$\{\Longrightarrow\}$ Partiamo dal valore

$$\sum_{i=1}^{m} d^{-l_i} = \sum_{x \in X} d^{-l_c(x)}.$$

e vediamo che valori assume la quantità

$$\left(\sum_{x\in X} d^{-l_c(x)}\right)^k.$$

Come possiamo riscrivere la potenza di una sommatoria? Vediamolo con un esempio.

Esempio 6.1: Se $k = 2$ allora

$$\left(\sum_i a_i\right)^2 = \sum_i a_i \sum_j a_j = \sum_i \sum_j a_i a_j.$$

Visto questo esempio, possiamo dire che

$$\begin{aligned}\left(\sum_{x\in X} d^{-l_c(x)}\right)^k &= \sum_{x_1\in X} \cdots \sum_{x_k\in X} d^{-l_c(x_1)} \cdot \ldots \cdot d^{-l_c(x_k)} = \\ &= \sum_{x_1\cdot\ldots\cdot x_k \in X^k} d^{-(l_c(x_1)+\ldots+l_c(x_k))} = \\ &= \sum_{x_1\cdot\ldots\cdot x_k \in X^k} d^{-l_{C_k}(x_1\cdot\ldots\cdot x_k)}.\end{aligned}$$

Cosa abbiamo dentro l'insieme X^k? Abbiamo tutte le possibili combinazioni (*in realtà disposizioni*) di k simboli, ma come sono le lunghezze di queste parole?

La minima è sicuramente k, e questo succede quando ad ogni simbolo $x_i \mid i = 1, \ldots, k$ assegno la lunghezza 1 (*in questo caso dovrei avere $m \leq d$ sennò il codice è singolare e non va bene*), mentre la massima è $kl_{\max}$, come abbiamo mostrato prima.

Andiamo a partizionare l'insieme X^k in insiemi X^k_t, ognuno dei quali contiene delle parole di k simboli lunghi in totale $t \geq 1$. In poche parole

$$X^k_t = \left\{(x_1, \ldots, x_k) \in X^k \mid l_{C_k}(x_1, \ldots, x_k) = t\right\}.$$

Nell'insieme $X^k_{kl_{\max}}$ abbiamo le parole di lunghezza massima. Riscriviamo la sommatoria come

$$\begin{aligned}\sum_{x_1\cdot\ldots\cdot x_k \in X^k} d^{-l_{C_k}(x_1\cdot\ldots\cdot x_k)} &= \sum_{n=1}^{kl_{\max}} \sum_{x_1\cdot\ldots\cdot x_k \in X^k_n} d^{-l_{C_k}(x_1\cdot\ldots\cdot x_k)} = \\ &= \sum_{n=1}^{kl_{\max}} \sum_{x_1\cdot\ldots\cdot x_k \in X^k_n} d^{-n} = \\ &= \sum_{n=1}^{kl_{\max}} \left|X^k_n\right| d^{-n}.\end{aligned}$$

La nostra funzione C_k è iniettiva (*non singolare*) perché il codice è UD, quindi il codominio di questo codice deve essere più grande o al massimo uguale al dominio. Il dominio è X_n^k, il codominio è D^n, quindi $\left|X_n^k\right| \leq |D^n| = d^n$, ma allora (*dominio e codominio un po' sus*)

$$\sum_{n=1}^{kl_{\max}} \left|X_n^k\right| d^{-n} \leq \sum_{n=1}^{kl_{\max}} d^n d^{-n} = \sum_{n=1}^{k_{\max}} 1 = kl_{\max}.$$

Siamo arrivati ad avere, dopo tutta sta catena, alla relazione

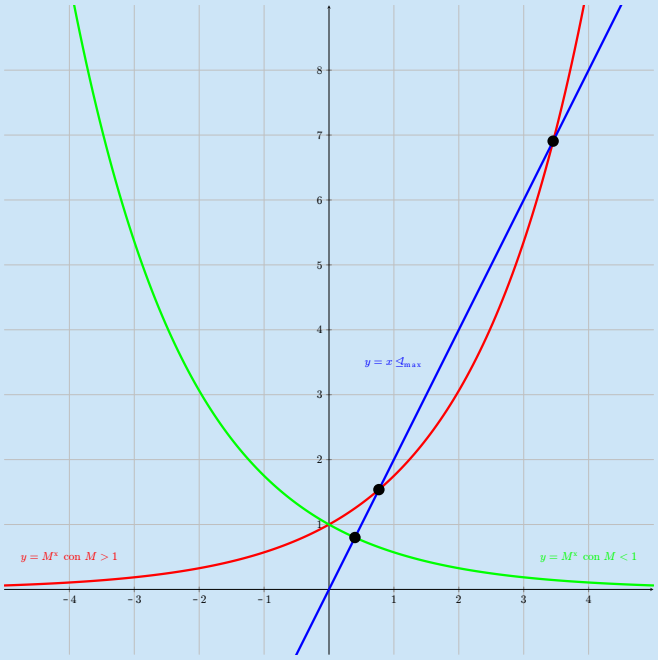
$$\left(\sum_{x \in X} d^{-l_c(x)}\right)^k \leq kl_{\max}.$$

Chiamando

$$M = \sum_{x \in X} d^{-l_c(x)},$$

dobbiamo verificare che

$$M^k \leq kl_{\max}.$$



Se disegno M^k ottengo due grafici diversi:

- se $M > 1$ ho un esponenziale crescente, che da un certo punto k_0 rende falsa la relazione;
- se $0 \leq M \leq 1$ ho un esponenziale decrescente, che da un certo punto k_0 rende vera la relazione.

Ma allora

$$M \leq 1 \implies \sum_{i=1}^{m} d^{-l_c(x_i)} \leq 1.$$

■

7. Parenti dell'entropia

L'**entropia congiunta** indica la quantità di informazione media che serve per comunicare due messaggi a partire dai simboli di una stessa sorgente. Data la sorgente $\langle X, p \rangle$ e date le variabili aleatorie $\mathbb{X}$ e$\mathbb{Y}$, essa è definita come

$$H(\mathbb{X}, \mathbb{Y}) = \sum_{x \in X} \sum_{y \in X} p(x, y) \log\left(\frac{1}{p(x, y)}\right).$$

L'**entropia condizionata** indica invece la quantità di informazione media che serve per comunicare un messaggio estratto da $\mathbb{Y}$ sapendo che ho già mandato un messaggio estratto da $\mathbb{X}$ dalla stessa sorgente. Data la sorgente $\langle X, p \rangle$ e date le variabili aleatorie $\mathbb{X}$ e$\mathbb{Y}$, essa è definita come

$$\begin{aligned} H(\mathbb{Y} \mid \mathbb{X}) &= \sum_{x \in X} p(x) H(\mathbb{Y} \mid \mathbb{X} = x) = \\ &= \sum_{x \in X} p(x) \left(\sum_{y \in X} p(y \mid x) \log\left(\frac{1}{p(y \mid x)}\right) \right) = \\ &= \sum_{x \in X} \sum_{y \in X} p(x) p(y \mid x) \log\left(\frac{1}{p(y \mid x)}\right). \end{aligned}$$

Sapendo che

$$p(y \mid x) = \frac{p(x, y)}{p(x)}$$

allora l'entropia condizionata vale

$$H(\mathbb{Y} \mid \mathbb{X}) = \sum_{x \in X} \sum_{y \in X} p(x, y) \log\left(\frac{1}{p(y \mid x)}\right).$$

Se $\mathbb{X} = f(\mathbb{Y})$ allora $H(\mathbb{X} \mid \mathbb{Y}) = 0$ perché la variabile $\mathbb{X}$ la posso ricavare facilmente dalla $\mathbb{Y}$.

Teorema 7.1 (*Chain rule per l'entropia*):

$$H(\mathbb{X}, \mathbb{Y}) = H(\mathbb{X}) + H(\mathbb{Y} \mid \mathbb{X}) = H(\mathbb{Y}) + H(\mathbb{X} \mid \mathbb{Y}).$$

Qua abbiamo bisogno della **probabilità marginale**: essa è definita come

$$p(x) = \sum_{y \in Y} p(x, y).$$

Dimostrazione 7.1: Verifichiamo che

$$
\begin{aligned}
H(\mathbb{X},\mathbb{Y}) &= \sum_{x\in X}\sum_{y\in X} p(x,y)\log\left(\frac{1}{p(x,y)}\right) = \\
&= \sum_{x\in X}\sum_{y\in X} p(x,y)\log\left(\frac{1}{p(x)}\cdot\frac{1}{p(y\mid x)}\right) = \\
&= \sum_{x\in X}\underbrace{\sum_{y\in X} p(x,y)}_{\text{congiunta}}\log\left(\frac{1}{p(x)}\right) + \sum_{x\in X}\sum_{y\in X} p(x,y)\log\left(\frac{1}{p(y\mid x)}\right) = \\
&= \sum_{x\in X} p(x)\log\left(\frac{1}{p(x)}\right) + H(\mathbb{Y}\mid\mathbb{X}) = H(\mathbb{X}) + H(\mathbb{Y}\mid\mathbb{X}).
\end{aligned}
$$

Lo stesso ragionamento lo possiamo fare invertendo $\mathbb{X}$ e $\mathbb{Y}$. ■

Questa regola vale anche negli spazi condizionati, ovvero

$$H(\mathbb{X},\mathbb{Y}\mid\mathbb{W}) = H(\mathbb{X}\mid\mathbb{W}) + H(\mathbb{Y}\mid\mathbb{X},\mathbb{W}).$$

Un altro parente dell'entropia, penso il prozio o il cugino, è l'**informazione mutua**: essa ci indica la quantità di informazione media che rilascia $\mathbb{Y}$ rispetto a $\mathbb{X}$. Si calcola con

$$I(\mathbb{X},\mathbb{Y}) = \sum_{x\in X}\sum_{y\in X} p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right).$$

È molto simile all'entropia relativa: con questa condivide la proprietà di essere ≥ 0, infatti non possiamo togliere informazioni da $\mathbb{X}$ avendo anche $\mathbb{Y}$. Inoltre, a differenza dell'entropia relativa, l'informazione mutua è simmetrica.

Lemma 7.1:

$$I(\mathbb{X},\mathbb{Y}) = H(\mathbb{X}) - H(\mathbb{X}\mid\mathbb{Y}).$$

Dimostrazione 7.2: Valutiamo

$$
\begin{aligned}
I(\mathbb{X},\mathbb{Y}) &= \sum_{x\in X}\sum_{y\in X} p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right) = \\
&= \sum_{x\in X}\sum_{y\in X} p(x,y)\log\left(\frac{p(y)p(x\mid y)}{p(x)p(y)}\right) = \\
&= \sum_{x\in X}\underbrace{\sum_{y\in X} p(x,y)}_{\text{marginale}}\log\left(\frac{1}{p(x)}\right) - \sum_{x\in X}\sum_{y\in X} p(x,y)\log\left(\frac{1}{p(x\mid y)}\right) = \\
&= \sum_{x\in X} p(x)\log\left(\frac{1}{p(x)}\right) - H(\mathbb{X}\mid\mathbb{Y}) = H(\mathbb{X}) - H(\mathbb{X}\mid\mathbb{Y}).
\end{aligned}
$$

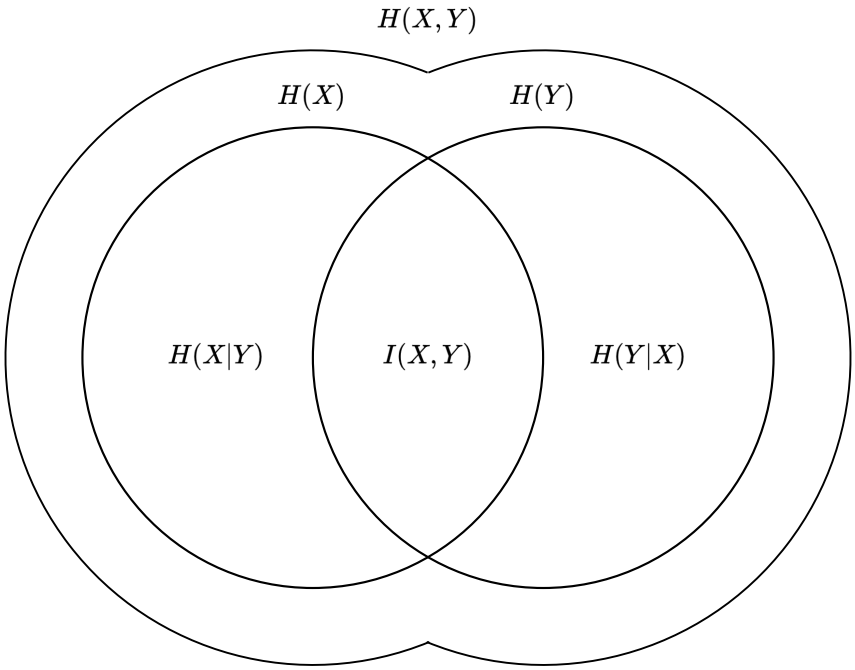
Lo stesso ragionamento lo possiamo fare invertendo $\mathbb{X}$ e $\mathbb{Y}$ (*credo*). ■

Se $\mathbb{X}$ e $\mathbb{Y}$ sono indipendenti allora $I(\mathbb{X}, \mathbb{Y}) = 0$ perché $\mathbb{Y}$ non rilascia informazioni su $\mathbb{X}$. Se invece le due variabili sono dipendenti allora $I(\mathbb{X}, \mathbb{Y}) = H(\mathbb{X})$ perché $\mathbb{X}$ è ricavabile totalmente da $\mathbb{Y}$.

Un modo forse più comodo di scrivere l'informazione mutua è

$$I(\mathbb{X}, \mathbb{Y}) = H(\mathbb{X}) - H(\mathbb{X} \mid \mathbb{Y}) = H(\mathbb{X}) - (H(\mathbb{X}, \mathbb{Y}) - H(\mathbb{Y})) = H(\mathbb{X}) + H(\mathbb{Y}) - H(\mathbb{X}, \mathbb{Y}),$$

che è molto simile alla probabilità dell'unione di due eventi.



Teorema 7.2 (*Data Processing Inequality*): Siano $\mathbb{X}, \mathbb{Y}, \mathbb{W}$ variabili aleatorie con codominio finito tali che $p(x, y, w)$ soddisfa

$$p(x, w \mid y) = p(x \mid y)p(w \mid y) \quad \forall x, y, w$$

ovvero x, w indipendenti dalla y condizionante. Allora

$$I(\mathbb{X}, \mathbb{Y}) \geq I(\mathbb{X}, \mathbb{W}).$$

Dimostrazione 7.3: Valutiamo

$$
\begin{aligned}
I(\mathbb{X},(\mathbb{Y},\mathbb{W})) &= \sum_{x,y,w\in X} p(x,y,w)\log\left(\frac{p(x,y,w)}{p(x)p(y,w)}\right) = \\
&= \text{scompongo con la congiunta al numeratore e al denominatore} = \\
&= \sum_{x,y,w\in X} p(x,y,w)\log\left(\frac{p(y\mid x,w)p(x,w)}{p(x)p(y\mid w)p(w)}\right) = \\
&= \sum_{x,w\in X}\underbrace{\sum_{y\in X} p(x,y,w)}_{\text{marginale}}\log\left(\frac{p(x,w)}{p(x)p(w)}\right) + \sum_{x,y,w\in X} p(x,y,w)\log\left(\frac{p(y\mid x,w)}{p(y\mid w)}\right) = \\
&= \sum_{x,w\in X} p(x,w)\log\left(\frac{p(x,w)}{p(x)p(w)}\right) + \sum_{x,y,w\in X} p(x,y,w)\log\left(\frac{p(x,y\mid w)}{p(y\mid w)p(x\mid w)}\right) = \\
&= I(\mathbb{X},\mathbb{W}) + I(\mathbb{X},\mathbb{Y}\mid\mathbb{W}).
\end{aligned}
$$

Questo vale per ogni terna $\mathbb{X},\mathbb{Y},\mathbb{W}$ quindi vale anche $I(\mathbb{X},(\mathbb{Y},\mathbb{W})) = I(\mathbb{X},\mathbb{Y}) + I(\mathbb{X},\mathbb{W}\mid\mathbb{Y})$.

Unendo i due risultati otteniamo

$$
I(\mathbb{X},\mathbb{Y}) + \underbrace{I(\mathbb{X},\mathbb{W}\mid\mathbb{Y})}_{=0} = I(\mathbb{X},\mathbb{W}) + \underbrace{I(\mathbb{X},\mathbb{Y}\mid\mathbb{W})}_{\geq 0},
$$

ma allora

$$
I(\mathbb{X},\mathbb{Y}) \geq I(\mathbb{X},\mathbb{W}). \qquad \blacksquare
$$

A cosa serve questo teorema? Abbiamo

$$
\mathbb{X} \overset{\text{rumore}}{\rightsquigarrow} \mathbb{Y} \overset{\text{algoritmo}}{\rightsquigarrow} \mathbb{W},
$$

ovvero sappiamo che $\mathbb{W}$ è indipendente da $\mathbb{X}$ e quindi il $\mathbb{W}$ processato non può essere più informativo di $\mathbb{Y}$ su $\mathbb{X}$.

Vediamo infine la **disuguaglianza di Fano** e cerchiamo di capire a cosa serve.

La sorgente manda un messaggio $\mathbb{X}$ e il ricevente riceve $\mathbb{Y}$. Se il canale non ha rumore nessun problema, ma se il canale distorce alcuni bit di informazione potrei ricevere un messaggio che è diverso da quello originale.

Usiamo quindi una funzione $g(\mathbb{Y})$ che cerca di risalire al simbolo $\mathbb{X}$ che è stato spedito. La funzione potrebbe azzeccare il vero simbolo che è stato spedito ma potrebbe anche darmi un'informazione sbagliata, ad esempio quando sono distorti tanti bit.

Chiamiamo p_e la probabilità che la funzione g sbagli a predire il simbolo $\mathbb{X}$ spedito, ovvero

$$
p_e = P(g(\mathbb{Y}) \neq \mathbb{X}).
$$

Ovviamente, più è alto il livello di distorsione del canale, più la funzione g sbaglia.

Teorema 7.3 (*Disuguaglianza di Fano*): Siano $\mathbb{X}, \mathbb{Y}$ due variabili casuali con valori in X, Y insiemi finiti. Sia $g : Y \longrightarrow X$ una funzione che mappa valori di Y in valori di X. Sia p_e la probabilità di errore quando uso $g(\mathbb{Y})$ per predire $\mathbb{X}$, ovvero $p_e = P(g(\mathbb{Y}) \neq \mathbb{X})$. Allora

$$p_e \geq \frac{H(\mathbb{X} \mid \mathbb{Y}) - 1}{\log_2(|X|)}.$$

Dimostrazione 7.4: Introduciamo una variabile bernoulliana $\mathbb{E}$ che indica il comportamento di g: quest'ultima o trova la $\mathbb{X}$ giusta, o trova una $\mathbb{X}$ sbagliata. In poche parole:

$$\mathbb{E} = \begin{cases} 1 & \text{se } g(\mathbb{Y}) \neq \mathbb{X} \\ 0 & \text{se } g(\mathbb{Y}) = \mathbb{X} \end{cases}.$$

Per la chain rule dell'entropia sappiamo che:

$$\begin{aligned} H(\mathbb{E}, \mathbb{X} \mid \mathbb{Y}) &\underset{\text{CR}}{=} H(\mathbb{E} \mid \mathbb{Y}) + H(\mathbb{X} \mid \mathbb{E}, \mathbb{Y}) = \\ &\underset{\text{CR}}{=} H(\mathbb{X} \mid \mathbb{Y}) + H(\mathbb{E} \mid \mathbb{X}, \mathbb{Y}). \end{aligned}$$

Consideriamo solo i due membri di destra, ovvero:

$$H(\mathbb{E} \mid \mathbb{Y}) + H(\mathbb{X} \mid \mathbb{E}, \mathbb{Y}) = H(\mathbb{X} \mid \mathbb{Y}) + H(\mathbb{E} \mid \mathbb{X}, \mathbb{Y})$$

Osserviamo che:

- $H(\mathbb{E} \mid \mathbb{Y}) \leq H(\mathbb{E})$ perché il condizionamento non aumenta l'entropia, lo vediamo dai diagrammi di Venn precedenti oppure osservando che il condizionamento può rilasciare o meno delle informazioni, ma sicuramente non ne toglie;
- $H(\mathbb{E} \mid \mathbb{X}, \mathbb{Y}) = 0$ perché conoscendo $\mathbb{X}$ e $\mathbb{Y}$ posso calcolare $g(\mathbb{Y})$ e vedere se è uguale a $\mathbb{X}$, e quindi sapere subito il valore di $\mathbb{E}$.

Siamo quindi nella seguente situazione:

$$H(\mathbb{E}) + H(\mathbb{X} \mid \mathbb{E}, \mathbb{Y}) \underset{\text{OSS1}}{\geq} H(\mathbb{E} \mid \mathbb{Y}) + H(\mathbb{X} \mid \mathbb{E}, \mathbb{Y}) = H(\mathbb{X} \mid \mathbb{Y}) + \underbrace{H(\mathbb{E} \mid \mathbb{X}, \mathbb{Y})}_{=0 \text{ per OSS2}}.$$

Consideriamo solo i due membri esterni, e quindi:

$$H(\mathbb{E}) + H(\mathbb{X} \mid \mathbb{E}, \mathbb{Y}) \geq H(\mathbb{X} \mid \mathbb{Y}).$$

Valutiamo la quantità

$$\begin{aligned} H(\mathbb{X} \mid \mathbb{E}, \mathbb{Y}) &= \sum_{e \in \mathbb{E}} p_e H(\mathbb{X} \mid \mathbb{E} = e, \mathbb{Y}) = \\ &= p(\mathbb{E} = 0) H(\mathbb{X} \mid \mathbb{E} = 0, \mathbb{Y}) + p(\mathbb{E} = 1) H(\mathbb{X} \mid \mathbb{E} = 1, \mathbb{Y}) = \\ &= (1 - p_e) \underbrace{H(\mathbb{X} \mid \mathbb{E} = 0, \mathbb{Y})}_{=0 \text{ perché ricavo da } g} + p_e H(\mathbb{X} \mid \mathbb{E} = 1, \mathbb{Y}) = \\ &= p_e H(\mathbb{X} \mid \mathbb{E} = 1, \mathbb{Y}). \end{aligned}$$

L'entropia è maggiorata dalla quantità $\log_2(|X|)$, ma se $\mathbb{E} = 1$ allora devo pescare da X tutti i valori tranne uno, quello corretto, perché appunto sto sbagliando a predire $\mathbb{X}$. Ma allora

$$p_e H(\mathbb{X} \mid \mathbb{E} = 0, \mathbb{Y}) \leq \log_2(|X| - 1).$$

Osserviamo infine che $H(\mathbb{E}) \leq 1$ perché $\mathbb{E}$ è una variabile bernoulliana.

Viste queste ultime due valutazioni abbiamo ottenuto:

$$1 + p_e \log_2(|X| - 1) \geq H(\mathbb{X} \mid \mathbb{Y}) \Longrightarrow p_e \geq \frac{H(\mathbb{X} \mid \mathbb{Y}) - 1}{\log_2(|X|)}. \qquad \blacksquare$$

Abbiamo mostrato che l'entropia è anche un **lower bound** per la probabilità di errore, oltre che per il valore atteso delle lunghezze del codice. Questo lower bound però è modulato sulla grandezza della sorgente, ma è comunque un lower bound.

Parte II — Teoria della Trasmissione

1. Canale

1.1. Definizione

Per ora ci siamo concentrati sulla prima parte del problema che aveva formulo Shannon, ovvero cercare di comprimere al massimo il messaggio da spedire sul canale. Ora ci concentriamo sul secondo problema, ovvero cercare di codificare il canale stesso per aggiungere ridondanza.

Un **canale** è definito dalla tupla

$$\langle X, Y, p(y \mid x) \rangle$$

formata da:

- X insieme dei simboli della sorgente;
- Y insieme dei simboli del ricevente;
- $p(y \mid x)$ probabilità di ottenere $y \in Y$ sapendo che è stato ricevuto $x \in X$. Questa probabilità è la **matrice di canale**, che ci consente di definire il comportamento del canale.

Noi useremo un **canale discreto senza memoria**:

- **discreto**: lavoriamo con i simboli della sorgente e del ricevente che sono un numero finito;
- **senza memoria**: sia $x^n = (x_1 \cdots x_n)$ un messaggio di n simboli di X e sia $y^n = (y_1 \cdots y_n)$ un messaggio di n simboli di Y. Osserviamo che, per la Chain Rule della probabilità, vale

  $$p(a, b, c) = p(a \mid b, c)p(b \mid c)p(c).$$

  Ma allora

  $$p(y^n \mid x^n) = p(y_n \mid y^{n-1}x^n)p(y_{n-1} \mid y^{n-2}x^n) \cdots p(y_3 \mid y^2 x^n)p(y_2 \mid y^1 x^n)p(y_1 \mid x^n).$$

  Il canale è *senza memoria*, ovvero il canale si disinteressa di quello che è stato spedito prima del carattere appena ricevuto e di quello che verrà spedito dopo il carattere appena ricevuto. Quindi:

  $$p(y^n \mid x^n) = p(y_n \mid x_n)p(y_{n-1} \mid x_{n-1}) \cdots p(y_1 \mid x_1) = \prod_{i=1}^{n} p(y_i \mid x_i).$$

Dobbiamo definire un'ultima cosa: la **capacità del canale**. Essa rappresenta la massima informazione che possiamo trasmettere quando accediamo al canale. Se immaginiamo un fiume, la capacità di quest'ultimo è la sua portata, ovvero quanta acqua passa nell'unità di tempo. In un canale, l'informazione è l'acqua e l'accesso al canale è l'unità di tempo.

La capacità di un canale $\langle X, Y, p(y \mid x) \rangle$ è definita come

$$C = \max_{p(x)} I(\mathbb{X}, \mathbb{Y}),$$

ovvero su tutte le distribuzioni di probabilità di $\mathbb{X}$ prendiamo l'informazione mutua massima.

Noi sappiamo che $I(\mathbb{X}, \mathbb{Y}) \geq 0$, ma anche che

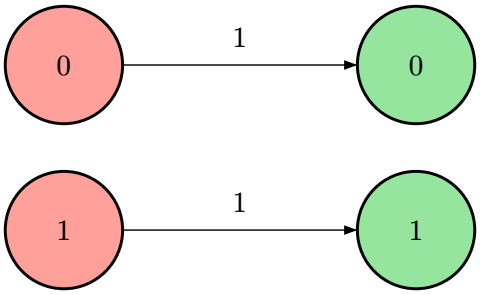
$$\begin{aligned} I(\mathbb{X}, \mathbb{Y}) &= H(\mathbb{X}) - H(\mathbb{X} \mid \mathbb{Y}) \\ &= H(\mathbb{Y}) - H(\mathbb{Y} \mid \mathbb{X}). \end{aligned}$$

Nella prima relazione $H(\mathbb{X}) \leq \log_2(|X|)$ quindi $H(\mathbb{X}) - H(\mathbb{X} \mid \mathbb{Y}) \leq \log_2(|X|)$. Possiamo fare un discorso analogo per la seconda relazione, ovvero $H(\mathbb{Y}) - H(\mathbb{Y} \mid \mathbb{X}) \leq \log_2(|Y|)$. Per far valere entrambe le relazioni prendiamo il minimo tra le due quantità. Ma allora

$$0 \leq C \leq \min(\log_2(|X|), \log_2(|Y|)).$$

1.2. Canale binario senza rumore

Un **canale binario senza rumore** è il canale più semplice che possiamo costruire.



Siano $\mathbb{X}, \mathbb{Y}$ due variabili casuali che pescano dagli insiemi X e Y. Essendo in un canale binario, abbiamo che

$$X = Y = \{0, 1\}.$$

Costruiamo la matrice $p(y \mid x)$ come

X/Y	0	1
0	1	0
1	0	1

.

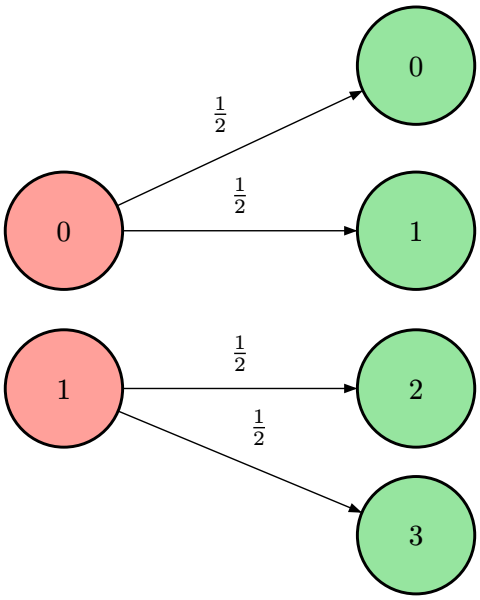
Calcoliamo l'informazione mutua con

$$I(\mathbb{X}, \mathbb{Y}) = H(\mathbb{X}) - H(\mathbb{X} \mid \mathbb{Y}) = H(\mathbb{X}).$$

La capacità C è il massimo di tutte le informazioni mutue, quindi il massimo delle entropie di $\mathbb{X}$, ma $\mathbb{X}$ è una bernoulliana che ha valore massimo 1, e quindi anche la capacità è 1.

1.3. Canale con rumore e uscite disgiunte

Un **canale con rumore e uscite disgiunte** è un esempio di canale che pesca da due insiemi diversi.



Siano $\mathbb{X}, \mathbb{Y}$ due variabili casuali che pescano dagli insiemi X e Y. In questo caso abbiamo

$$X = \{0, 1\} \quad | \quad Y = \{0, 1, 2, 3\}.$$

Costruiamo la matrice $p(y \mid x)$ come

X/Y	0	1	2	3
0	0.5	0.5	0	0
1	0	0	0.5	0.5

.

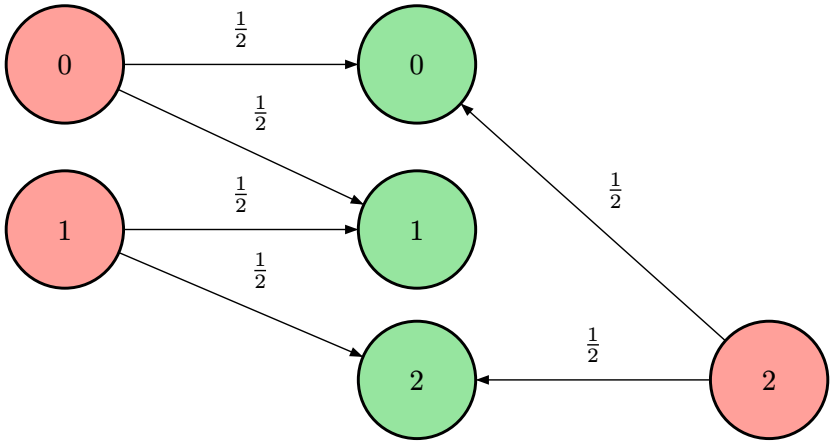
Calcoliamo la capacità del canale come

$$C = \max_{p(x)} I(\mathbb{X}, \mathbb{Y}) = \max_{p(x)} H(\mathbb{X}) - H(\mathbb{X} \mid \mathbb{Y}).$$

La seconda quantità è 0 perché sapendo $\mathbb{Y}$ sappiamo anche la $\mathbb{X}$. Dobbiamo quindi massimizzare $H(\mathbb{X})$, ma questa è una bernoulliana che ha valore massimo 1, quindi la capacità è 1.

1.4. Macchina da scrivere rumorosa

La **macchina da scrivere rumorosa** è un canale lievemente più complesso di quelli visti per ora.



Siano $\mathbb{X}, \mathbb{Y}$ due variabili casuali che pescano dagli insiemi X e Y. In questo caso abbiamo

$$X = Y = \{0, 1, 2\}.$$

Costruiamo la matrice $p(y \mid x)$ come

X/Y	0	1	2
0	0.5	0.5	0
1	0	0.5	0.5
2	0.5	0	0.5

.

Calcoliamo la capacità del canale come

$$C = \max_{p(x)} I(\mathbb{X}, \mathbb{Y}) = \max_{p(x)} H(\mathbb{Y}) - H(\mathbb{Y} \mid \mathbb{X}).$$

Valutiamo la seconda quantità come

$$\begin{aligned} H(\mathbb{Y} \mid \mathbb{X}) &= p(\mathbb{X} = 0) H(\mathbb{Y} \mid \mathbb{X} = 0) + p(\mathbb{X} = 1) H(\mathbb{Y} \mid \mathbb{X} = 1) + p(\mathbb{X} = 2) H(\mathbb{Y} \mid \mathbb{X} = 2) = \\ &= p_0 H(\mathbb{Y} \mid \mathbb{X} = 0) + p_1 H(\mathbb{Y} \mid \mathbb{X} = 1) + p_2 H(\mathbb{Y} \mid \mathbb{X} = 2). \end{aligned}$$

Non conosciamo p_0, p_1, p_2 ma sappiamo che l'entropia viene massimizzata quando le probabilità dei simboli sono distribuite uniformemente, quindi assumiamo l'uniformità

$$p_0 = p_1 = p_2 = \frac{1}{3}.$$

Calcoliamo ogni entropia come

$$\forall i \in \{0,1,2\} \quad H(\mathbb{Y} \mid \mathbb{X} = i) = \sum_{k=0}^{2} p(y_k \mid \mathbb{X} = i) \log_2 \left(\frac{1}{p(y_k \mid \mathbb{X} = i)} \right).$$

Non abbiamo voglia di fare i conti, quindi qualcuno ci dice che le entropie valgono 1. Ma allora

$$H(\mathbb{Y} \mid \mathbb{X}) = p_0 + p_1 + p_2 = 1.$$

Rimane solo $H(\mathbb{Y})$. Scegliamo una distribuzione uniforme che massimizzi l'entropia, quindi

$$p(\mathbb{Y} = 0) = p(\mathbb{Y} = 1) = p(\mathbb{Y} = 2) \underset{\text{TPT}}{=} \left(\frac{1}{3} \cdot \frac{1}{2} \right) + \left(\frac{1}{3} \cdot \frac{1}{2} \right) = \frac{1}{3}.$$

L'entropia diventa quindi

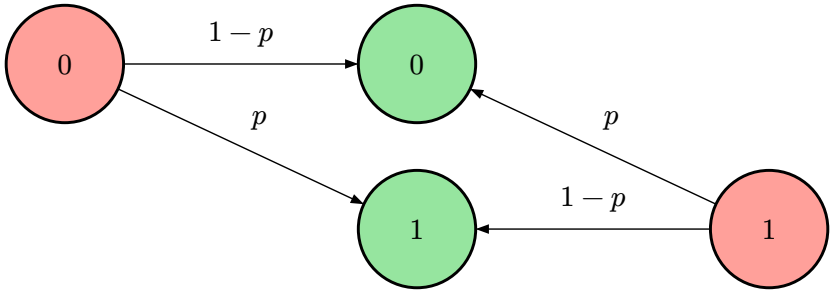
$$H(\mathbb{Y}) = \sum_{i=0}^{2} p(y_i) \log_2 \left(\frac{1}{p(y_i)} \right) = 3 \cdot \frac{1}{3} \log_2(3) = \log_2(3).$$

La capacità è quindi

$$C = \log_2(3) - 1 \approx 0.585.$$

1.5. Canale binario simmetrico

Vediamo ora due canali ancora più complessi: partiamo dal **canale binario simmetrico**.



Siano $\mathbb{X}, \mathbb{Y}$ due variabili casuali che pescano dagli insiemi X e Y. In questo caso abbiamo

$$X = Y = \{0,1\}.$$

Costruiamo la matrice $p(y \mid x)$ come

X/Y	0	1
0	$1-p$	p
1	p	$1-p$

.

Calcoliamo la capacità del canale come

$$C = \max_{p(x)} I(\mathbb{X}, \mathbb{Y}) = \max_{p(x)} H(\mathbb{Y}) - H(\mathbb{Y} \mid \mathbb{X}).$$

Valutiamo la seconda quantità come

$$H(\mathbb{Y} \mid \mathbb{X}) = \sum_{i=0}^{1} p(\mathbb{X} = i) H(\mathbb{Y} \mid \mathbb{X} = i).$$

Calcoliamo $H(\mathbb{Y} \mid \mathbb{X} = i)$ per $i \in \{0, 1\}$:

$$\begin{aligned} H(\mathbb{Y} \mid \mathbb{X} = 0) &= p(\mathbb{Y} = 0 \mid \mathbb{X} = 0) \log_2 \left(\frac{1}{p(\mathbb{Y} = 0 \mid \mathbb{X} = 0)} \right) + \\ &+ p(\mathbb{Y} = 1 \mid \mathbb{X} = 0) \log_2 \left(\frac{1}{p(\mathbb{Y} = 1 \mid \mathbb{X} = 0)} \right) = \\ &= (1 - p) \log_2 \left(\frac{1}{1 - p} \right) + p \log_2 \left(\frac{1}{p} \right) = H(p) \end{aligned}$$

$$H(\mathbb{Y} \mid \mathbb{X} = 1) = \text{idem con patate} = H(p).$$

In poche parole, so che $H(\mathbb{Y} \mid \mathbb{X} = i)$ vale come l'entropia di una bernoulliana di parametro p, che però non conosco in questo momento. Ma allora

$$H(\mathbb{Y} \mid \mathbb{X}) = p(\mathbb{X} = 0) H(p) + p(\mathbb{X} = 1) H(p) = (p(\mathbb{X} = 0) + p(\mathbb{X} = 1)) H(p) = H(p).$$

Il calcolo di $H(\mathbb{Y})$ ricade nuovamente sull'entropia di una variabile distribuita uniformemente. Calcoliamo le $p(\mathbb{Y})$ singolarmente:

$$p(\mathbb{Y} = 0) = p(\mathbb{X} = 0) p(\mathbb{Y} = 0 \mid \mathbb{X} = 0) + p(\mathbb{X} = 1) p(\mathbb{Y} = 0 \mid \mathbb{X} = 1) = \frac{1}{2}(1 - p) + \frac{1}{2} p = \frac{1}{2}$$

$$p(\mathbb{Y} = 1) = \text{idem con patate} = \frac{1}{2}.$$

Ma allora

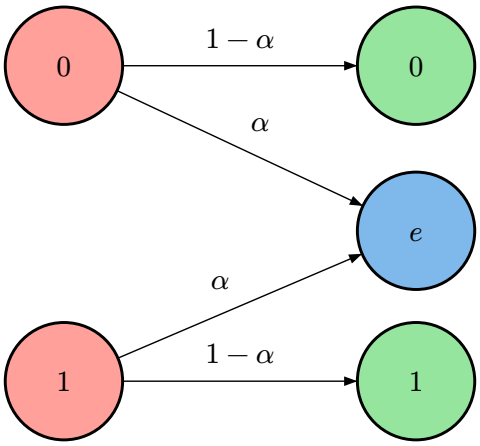
$$H(\mathbb{Y}) = \sum_{i=0}^{1} p(y_i) \log_2 \left(\frac{1}{p(y_i)} \right) = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 1 = 1.$$

La capacità è quindi

$$C = 1 - H(p).$$

1.6. Canale binario a cancellazione

Terminiamo la carrellata di canali con il **canale binario a cancellazione**.



Siano $\mathbb{X}, \mathbb{Y}$ due variabili casuali che pescano dagli insiemi X e Y. Essendo in un canale binario, abbiamo che

$$X = Y = \{0, 1\}.$$

Costruiamo la matrice $p(y \mid x)$ come

$$\begin{array}{c|ccc} X/Y & 0 & 1 & e \\ \hline 0 & 1-\alpha & 0 & \alpha \\ 1 & 0 & 1-\alpha & \alpha \end{array} \quad .$$

Calcoliamo la capacità del canale come

$$C = \max_{p(x)} I(\mathbb{X}, \mathbb{Y}) = \max_{p(x)} H(\mathbb{Y}) - H(\mathbb{Y} \mid \mathbb{X}).$$

Osserviamo che la seconda quantità, visto che $\mathbb{X}$ è una variabile bernoulliana di parametro α, si comporta come la quantità del canale precedente, quindi $H(\mathbb{Y} \mid \mathbb{X}) = H(\alpha)$.

Introduciamo una variabile $\mathbb{W}$ tale che

$$\mathbb{W} = \begin{cases} 1 & \text{se } \mathbb{Y} = e \\ 0 & \text{altrimenti} \end{cases} \quad .$$

Sappiamo, per la Chain Rule dell'entropia, che

$$H(\mathbb{Y}, \mathbb{W}) = H(\mathbb{Y}) + H(\mathbb{W} \mid \mathbb{Y}) = H(\mathbb{W}) + H(\mathbb{Y} \mid \mathbb{W}).$$

Consideriamo l'uguaglianza di destra e isoliamo $H(\mathbb{Y})$ ottenendo

$$H(\mathbb{Y}) = H(\mathbb{W}) + H(\mathbb{Y} \mid \mathbb{W}) - H(\mathbb{W} \mid \mathbb{Y}).$$

Notiamo che $H(\mathbb{W} \mid \mathbb{Y}) = 0$ perché se conosco $\mathbb{Y}$ posso dire con precisione il valore di $\mathbb{W}$.

Abbiamo tolto un fattore. Calcoliamo ora $H(\mathbb{W})$ con la formula classica di entropia, calcolando però prima le singole probabilità $p(\mathbb{W} = i)$ come

$$\begin{aligned} p(\mathbb{W} = 1) &= p(\mathbb{X} = 0)p(\mathbb{W} = 1 \mid \mathbb{X} = 0) + p(\mathbb{X} = 1)p(\mathbb{W} = 1)p(\mathbb{W} = 1 \mid \mathbb{X} = 1) = \\ &= p(\mathbb{X} = 0)\alpha + p(\mathbb{X} = 1)\alpha = \alpha(p(\mathbb{X} = 0) + p(\mathbb{X} = 1)) = \alpha \\ p(\mathbb{W} = 0) &= 1 - p(\mathbb{W} = 1) = 1 - \alpha. \end{aligned}$$

Ma allora

$$H(\mathbb{W}) = \sum_{i=0}^{1} p(w_i) \log_2\left(\frac{1}{p(w_i)}\right) = (1-\alpha)\log_2\left(\frac{1}{1-\alpha}\right) + \alpha \log_2\left(\frac{1}{\alpha}\right) = H(\alpha).$$

Calcoliamo infine $H(\mathbb{Y} \mid \mathbb{W})$ come

$$\begin{aligned} H(\mathbb{Y} \mid \mathbb{W}) &= p(\mathbb{W} = 0)H(\mathbb{Y} \mid \mathbb{W} = 0) + p(\mathbb{W} = 1)H(\mathbb{Y} \mid \mathbb{W} = 1) = \\ &= (1-\alpha)H(\mathbb{X}) + \alpha \cdot 0 = H(\mathbb{X})(1-\alpha). \end{aligned}$$

Questo vale perché:

- se $\mathbb{W} = 0$ vuol dire che non ho errore, ma se non ho errori quello che mando è quello che ricevo, quindi dipende solo dalla sorgente;
- se $\mathbb{W} = 1$ vuol dire che ho avuto errore e che conosco già il valore di $\mathbb{Y}$.

Possiamo calcolare, **FINALMENTE**, il valore di $H(\mathbb{Y})$ come

$$H(\mathbb{Y}) = H(\alpha) + H(\mathbb{X})(1-\alpha).$$

La capacità del canale è quindi

$$C = \max_{p(x)} H(\mathbb{Y}) - H(\mathbb{Y} \mid \mathbb{X}) = H(\alpha) + H(\mathbb{X})(1-\alpha) - H(\alpha) = H(\mathbb{X})(1-\alpha) = 1 - \alpha$$

perché $H(\mathbb{X})$ è l'entropia di una bernoulliana, che ha valore massimo 1.

2. Secondo teorema di Shannon

Il canale che abbiamo ora non ci va molto bene: come nella parte di codifica, definiremo una sorta di estensione del canale per poter poi *«spalmare»* l'errore quando l'estensione diventa molto grande.

Definiamo quindi l'**estensione** n-esima del canale come la tupla

$$C_n = \langle X^n, Y^n, p(y^n \mid x^n) \rangle$$

tale che

$$p(y^n \mid x^n) = \prod_{i=1}^{n} p(y_i \mid x_i)$$

per indipendenza.

Un **codice canale** per il canale C_n è di tipo (M, n) se:

- M è il massimo numero di messaggi che voglio codificare/spedire sul canale, ovvero è l'insieme di messaggi $\{1, ..., M\}$;
- n è il numero di accessi che facciamo al canale;
- x^n è la funzione di codifica tale che $x^n : \{1, ..., M\} \longrightarrow X^n$;
- g è la funzione di decodifica tale che $y^n : \mathbb{Y}^n \longrightarrow \{1, ..., M\}$.

Manca ancora qualcosa, ma per sapere cosa vediamo cosa stiamo facendo ora:

$$M \xrightarrow{x^n} X^n \xrightarrow{\text{send}} (\text{canale con rumore}) \xrightarrow{\text{send}} Y^n.$$

Siamo in Y^n, abbiamo a disposizione la funzione g di decodifica, che però potrebbe fare errori: questo è dato dal fatto che siamo passati in un canale con rumore. Dobbiamo quindi applicare g e indovinare il risultato di questa decodifica.

Completiamo il nostro codice canale con:

- λ_i, definita come $P(g(y^n) \neq i \mid \mathbb{X}^n = x^n(i))$, ovvero la probabilità che la funzione g decodifichi y^n in un valore diverso da i, che è il nostro messaggio di partenza che è stato codificato da x^n;
- $\lambda^{(n)}$, definita come $\max_{i=1,...,n} \lambda_i$, ovvero la massima probabilità di errore; quantità comoda come upper bound alle probabilità di errore;
- $p_e^{(n)}$, definita come $\frac{1}{n} \sum_{i=1}^{n} \lambda_i$; quantità comoda invece per gestire delle probabilità medie.

Vale ovviamente $p_e^{(n)} \leq \lambda^{(n)}$.

Definiamo il **tasso di trasmissione** di un codice di tipo (M, n) come la quantità

$$R = \frac{\log_2(M)}{n}.$$

Nel nostro caso specifico, codificando in binario e usando n accessi al canale, il numero di messaggi disponibili è 2^n, ma allora

$$R = \frac{\log_2(2^n)}{n} = 1.$$

Questo caso è quello *utopico*, ovvero quando il canale è senza rumore. Se quest'ultimo è presente il tasso di trasmissione è ovviamente minore.

Nella realtà viene usato il **tasso di trasmissione raggiungibile**. Un tasso di trasmissione R è **raggiungibile** se esiste una sequenza di codici canale di tipo $\left(2^{nR}, n\right) \mid n = 1, 2, ...$ tale che

$$\lim_{n\to\infty} \lambda^{(n)} = 0.$$

Avendo $R < 1$ il valore di M del codice canale viene minore del caso utopico. Possiamo vedere R come la quantità che ci indica quanto sfoltire il 2^n teorico per non avere errori. Ovviamente, prenderemo la parte intera della quantità 2^{nR} perché dobbiamo codificare un numero intero di messaggi.

Vediamo due proprietà, una statistica e uno viscontea, che ci dicono la stessa cosa.

Teorema 2.1 (*Legge dei grandi numeri*): Per ogni sequenza $\mathbb{X}_1, ..., \mathbb{X}_n$ di variabili casuali i.i.d con valore atteso μ finito, allora

$$\forall \varepsilon > 0 \quad \lim_{n\to\infty} P\left(\left|\frac{1}{n}\sum_{i=1}^{n} \mathbb{X}_i - \mu\right| > \varepsilon\right) = 0.$$

Questa legge afferma che, usando la media campionaria come stimatore per il valore atteso delle variabili casuali $\mathbb{X}_i$, la probabilità di commettere un errore maggiore ε è nulla se consideriamo un buon numero di variabili casuali.

Una proprietà analoga è quella di **equipartizione asintotica**.

Teorema 2.2 (*AEP*): Per ogni sequenza $\mathbb{X}_1, ..., \mathbb{X}_n$ di variabili casuali i.i.d con entropia $H(\mathbb{X})$ finita, allora

$$\forall \varepsilon > 0 \quad \lim_{n\to\infty} P\left(\left|\frac{1}{n}\log_2\left(\frac{1}{p(x_1)\cdot ... \cdot p(x_n)}\right) - H(\mathbb{X})\right| > \varepsilon\right) = 0.$$

La proprietà è analoga alla legge dei grandi numeri, solo che si basa sull'entropia. Inoltre, questa proprietà vale per ogni variabile casuale $\mathbb{X}$ sulla quale calcoliamo l'entropia $H(\mathbb{X})$.

Diamo ora una definizione di **insieme tipico**. Vediamo prima una definizione informale.

Peschiamo n oggetti creando una sequenza $(x_1, ..., x_n)$. Visto che le pescate sono i.i.d., la probabilità di pescare tale sequenza è il prodotto delle singole probabilità. Se questo prodotto è compreso tra due quantità che ora vedremo allora tale sequenza appartiene all'insieme tipico.

Vediamo ora la definizione formale. L'insieme tipico è l'insieme

$$A_\varepsilon^{(n)} = \left\{(x_1, ..., x_n) \in X^n \mid 2^{-n(H(\mathbb{X})+\varepsilon)} \leq \prod_{i=1}^{n} p(x_i) \leq 2^{-n(H(\mathbb{X})-\varepsilon)}\right\}.$$

Facciamo vedere che l'insieme tipico è, in realtà, l'insieme formato da tutte le sequenze che rispettano la proprietà di equipartizione asintotica. Infatti:

$$2^{n(H(\mathbb{X})-\varepsilon)} \leq \frac{1}{\prod_{i=1}^{n} p(x_i)} \leq 2^{n(H(\mathbb{X})+\varepsilon)}$$

$$n(H(\mathbb{X})-\varepsilon) \leq \log_2\left(\frac{1}{\prod_{i=1}^{n} p(x_i)}\right) \leq n(H(\mathbb{X})+\varepsilon)$$

$$H(\mathbb{X})-\varepsilon \leq \frac{1}{n}\log_2\left(\frac{1}{\prod_{i=1}^{n} p(x_i)}\right) \leq H(\mathbb{X})+\varepsilon$$

$$-\varepsilon \leq \frac{1}{n}\log_2\left(\frac{1}{\prod_{i=1}^{n} p(x_i)}\right) - H(\mathbb{X}) \leq \varepsilon$$

$$\left|\frac{1}{n}\log_2\left(\frac{1}{\prod_{i=1}^{n} p(x_i)}\right) - H(\mathbb{X})\right| \leq \varepsilon$$

Dentro AEP abbiamo in realtà $< \varepsilon$, ma questo non cambia niente. Infatti, possiamo scrivere l'insieme tipico come

$$A_\varepsilon^{(n)} = \left\{(x_1, ..., x_n) \in X^n \mid \left|\frac{1}{n}\log_2\left(\frac{1}{\prod_{i=1}^{n} p(x_i)}\right) - H(\mathbb{X})\right| \leq \varepsilon\right\}.$$

Vista questa definizione, possiamo affermare che

$$\lim_{n\to\infty} P\left(A_\varepsilon^{(n)}\right) = 1$$

o che

$$\lim_{n\to\infty} P\left(\overline{A}_\varepsilon^{(n)}\right) = 0.$$

Il significato operativo di quanto detto fin'ora è il seguente: asintoticamente le sequenze che stanno nell'insieme tipico hanno tutte la stessa probabilità (*visto che le estrazioni sono i.i.d.*) ed è uguale al prodotto delle probabilità singole. Ma queste quantità sono limitate dai due bound a meno di ε, ma allora le probabilità sono tutte uguali a $2^{-nH(\mathbb{X})}$ a meno di ε. In poche parole:

$$P(x_1, ..., x_n) = \prod_{i=1}^{n} p(x_i) = \begin{cases} 0 & \text{se } (x_1, ..., x_n) \notin A_\varepsilon^{(n)} \\ 2^{-nH(\mathbb{X})} & \text{altrimenti} \end{cases}.$$

Teorema 2.3: Siano $\mathbb{X}_1, ..., \mathbb{X}_n$ delle variabili casuali i.i.d. e sia $A_\varepsilon^{(n)}$ l'insieme tipico ad esse associato. Allora:

1. possiamo dare un upper bound al numero di elementi dell'insieme tipico, ovvero
$$\forall n \in \mathbb{N} \quad \left|A_\varepsilon^{(n)}\right| \leq 2^{n(H(\mathbb{X})+\varepsilon)};$$
2. possiamo dare un lower bound al numero di elementi dell'insieme tipico, ovvero
$$\exists n_0 \in \mathbb{N} \mid \forall n > n_0 \quad \left|A_\varepsilon^{(n)}\right| \geq (1-\varepsilon)2^{n(H(\mathbb{X})-\varepsilon)}.$$

Vediamo l'insieme tipico in un altra versione. Supponiamo di trasmettere un messaggio $x^n \in X^n$ in un canale rumoroso, che quindi distorce l'informazione trasmessa. Questo messaggio verrà mappato nei messaggi che stanno nell'insieme tipico se riceviamo un messaggio che non è distante dalla parola

effettiva. Se invece siamo distanti dalla parola effettiva, non andremo a finire nell'insieme tipico. In poche parole, ogni messaggio $x^n \in X^n$ va a definire un insieme tipico nell'insieme Y^n. Tutti questi insiemi formano delle **bolle** dentro Y^n.

Ogni bolla ha una dipendenza da ciò che abbiamo spedito, ovvero dobbiamo considerare degli insiemi che sono grandi

$$\left|A_\varepsilon^{(n)}\right| \approx 2^{nH(\mathbb{Y} \mid \mathbb{X})}.$$

Vogliamo ovviamente che tutti questi insiemi siano disgiunti, così da poter decodificare senza sovrapposizioni di bolle di messaggi diverse. Prendiamo quindi una parte di questi messaggi, ovvero

$$M = \frac{2^{nH(\mathbb{Y})}}{2^{nH(\mathbb{Y} \mid \mathbb{X})}} = 2^{n(H(\mathbb{Y})-H(\mathbb{Y} \mid \mathbb{X}))} = 2^{nI(\mathbb{X},\mathbb{Y})}.$$

ASSURDO! Possiamo quindi riscrivere il tasso di trasmissione R come

$$R = \frac{\log_2(M)}{n} = \frac{\log_2\left(2^{nI(\mathbb{X},\mathbb{Y})}\right)}{n} = \frac{nI(\mathbb{X},\mathbb{Y})}{n} = I(\mathbb{X},\mathbb{Y}).$$

Definiamo quindi la nuova versione come l'**insieme congiuntamente tipico**. Esso è l'insieme

$$B_\varepsilon^{(n)} = \left\{ (x^n, y^n) \in X^n \times Y^n \mid \left| \frac{1}{n} \log_2 \left(\frac{1}{p(x_1,y_1) \cdot \ldots \cdot p(x_n,y_n)} \right) - H(\mathbb{X},\mathbb{Y}) \right| < \varepsilon \right\}.$$

Questo insieme gode di due **proprietà**:

- come per l'insieme tipico classico,

$$\lim_{n\to\infty} P\left(B_\varepsilon^{(n)}\right) = 1;$$

- se:
  - $\mathbb{X}_n$ ha distribuzione $p(\mathbb{X}_n = x^n) = \prod_i p(x_i)$ con $p(x_i)$ marginale di X e $\mathbb{X}_n$ sequenze tipiche di X;
  - $\mathbb{Y}_n$ ha distribuzione $p(\mathbb{Y}_n = y^n) = \prod_i p(y_i)$ con $p(y_i)$ marginale di Y e $\mathbb{Y}_n$ sequenze tipiche di Y;

  allora

$$\forall n \geq 1 \quad P\left((\mathbb{X}_n, \mathbb{Y}_n) \in B_\varepsilon^{(n)}\right) \leq 2^{-n(I(\mathbb{X},\mathbb{Y})-3\varepsilon)}.$$

In poche parole, l'insieme $B_\varepsilon^{(n)}$ contiene tutte le coppie di sequenze (x^n, y^n) che rispettano tre condizioni:

- le sequenze di $\mathbb{X}_n$ sono tipiche rispetto all'entropia $H(\mathbb{X})$;
- le sequenze di $\mathbb{Y}_n$ sono tipiche rispetto all'entropia $H(\mathbb{Y})$;
- le coppie di sequenze (x^n, y^n) sono tipiche rispetto all'entropia $H(\mathbb{X},\mathbb{Y})$.

Stiamo rispettando sia le distribuzioni marginali sia quella congiunta.

Le due proprietà ci dicono che per n grande quasi tutte le coppie osservabili di sequenze appartengono all'insieme $B_\varepsilon^{(n)}$. Inoltre, la probabilità di uscire da questo insieme è bassa e decresce esponenzialmente al crescere di n. Lavorando con l'insieme tipico abbiamo una rappresentazione affidabile e precisa del comportamento delle sequenze originali.

Possiamo vedere infine il secondo teorema di Shannon, madonna.

Teorema 2.4 (*Secondo teorema di Shannon*): Se $\langle X, Y, p(y \mid x)\rangle$ è un canale con capacità C, allora

$$\forall R < C \quad \exists K_1, ..., K_n \text{ di tipo } \left(2^{nR_n}, n\right) \mid \lim_{n\to\infty} R_n = R \wedge \lim_{n\to\infty} \lambda^{(n)}(K_n) = 0.$$

In altre parole, è possibile scegliere un codice che si avvina al massimo della capacità del canale tramite R, ovvero avvicinando il codice al massimo valore che R può assumere. Inoltre, così facendo, almeno teoricamente, si *"spalma"* l'errore sull'informazione trasmessa (*che è però massima*), perciò all'aumentare di n si riduce al minimo l'errore trasmesso. Ricordiamo anche che, all'aumentare di n, le *"sfere"* dell'insieme tipico tendono a non toccarsi permettendo di decodificare senza sovrapposizioni.

Parte III — Codici

1. Introduzione matematica

Partiamo dai numeri interi $\mathbb{Z}$: esso è un **insieme infinito**, ma che non ci piace tanto perché la nostra macchina fa i conti con i numeri finiti.

Trasformiamo $\mathbb{Z}$ nell'insieme $\mathbb{Z}_n$ con la relazione di equivalenza della **congruenza**, che contiene i numeri interi modulo n e quindi è un **insieme finito**.

Lavorando su n non ci va tanto bene perché in generale siamo su un **anello**, nel quale non tutti gli elementi hanno inverso. Questo è un problema, non posso fare le divisioni non avendo l'inverso, che è roba da pazzi se pensiamo di non poter dividere per 2 in un calcolatore. Passiamo quindi a $\mathbb{Z}_p$, che è definito in modo analogo a $\mathbb{Z}_n$ solo che p è un numero primo e quindi questo insieme è un **campo**, che mi garantisce la presenza di un inverso per ogni elemento dell'insieme.

Un calcolatore lavora con i byte, quindi ho 2^8 possibili valori, ma questo è ancora un problema perché 2^8 non è primo. Risolviamo scrivendo il byte come un **polinomio** a coefficienti in $\mathbb{Z}_p$, creando l'insieme $\mathbb{Z}_p[x]$.

Siamo nella stessa situazione di prima: sono ancora con un insieme infinito. Infatti, il grado di questi polinomi può anche essere infinito, quindi diamo un tetto al grado facendo il modulo con un polinomio, creando l'insieme finito $\mathbb{Z}_p[x] \bmod m(x)$.

Infine, ultimo problema che risolviamo, è quello dell'anello. Stiamo quozientando su un polinomio random, quindi non tutti gli elementi siamo sicuri abbiano un inverso. Per risolvere quest'ultimo problema cambiamo $m(x)$ in $p(x)$ **polinomio irriducibile** sul campo, e otteniamo l'insieme finale $\mathbb{Z}_p[x] \bmod p(x)$.

Quest'ultimo insieme è esattamente il **campo di Galois**. Un campo di Galois $\mathrm{GF}(p^m)$ è un campo che contiene un numero di elementi uguale a p^m e ognuno di questi è un polinomio che grado massimo $m-1$ a coefficienti in $\mathbb{Z}_p$. Il valore m indice anche il grado di $p(x)$.

2. Codici semplici

2.1. Introduzione

Il **rumore** è l'entità che distorce le informazioni che passano sul canale. Noi vogliamo usare dei codici che rilevino gli errori e, non sarebbe male, che li correggano anche.

Noi considereremo il **rumore bianco**. Esso ha le seguenti due proprietà:

- il rumore arriva in maniera indipendente sui bit, cioè la probabilità di avere un bit errato non dipende dalle probabilità di trovare errori nelle altre posizioni. Nella realtà la situazione è opposta: gli errori in una zona mi indicano grande probabilità di averne altri nelle vicinanze;
- la probabilità di avere un bit errato è fissata a p e questa quantità è uguale per tutte le posizioni che stiamo considerando. Nella realtà, ovviamente, è totalmente diverso: la probabilità di errore potrebbe variare nel tempo.

2.2. Bit di parità

Vediamo il buon vecchio **bit di parità**. Questo bit si calcola sommando tutti i bit del numero e la si fa $\bmod 2$. In poche parole, date $n-1$ cifre, il bit di parità lo calcoliamo come

$$x_n = \sum_{i=1}^{n-1} x_i \bmod 2.$$

Come facciamo a capire se abbiamo un problema sul canale e dobbiamo rispedire il messaggio? Prendiamo i dati ricevuti, calcoliamo il bit di parità e vediamo se otteniamo 0.

Questo codice viene usato per rilevare l'errore, ma non per correggerlo. Inoltre, se l'errore si spalma su un numero pari di bit, il bit di parità non riesce nemmeno a rilevare l'errore.

2.3. Codice ASCII

Vediamo il **codice ASCII**. Eh si fratello, il codice ASCII è un codice, assurdo.

Come funziona questo codice? Esso utilizza 7 bit per codificare un carattere e un bit di parità, posto in testa della codifica, molto facile.

2.4. Burst di errori

Vediamo un **burst** di errori, ovvero una zona di errori definita da un inizio e una fine. Faremo il controllo usando il bit di parità.

Esempio 2.4.1: Prendiamo il messaggio «Hello NCTU». Aggiungiamo un **carattere di parità**, non più un bit, che mettiamo alla fine. Questo carattere fa check di correttezza. Divido il messaggio in caratteri, scrivo in binario e poi calcolo i singoli bit di parità facendo il controllo di parità sulla colonna.

Otteniamo

$$\begin{pmatrix} H & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ e & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ l & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ l & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ o & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ N & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ C & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ T & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ U & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ \blacksquare & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

La codifica dell'ultima lettera è n.

Se so che l'errore inizia nelle ultime due colonne della prima riga e finisce nella prime colonne della seconda riga, potrei riuscire a sistemare. Il problema arriva se il burst copre più righe, perché più errori potrebbero annullarsi e quindi non venire rilevati.

2.5. Somma pesata

Sto spedendo un messaggio di n caratteri. I **codici a somma pesata** richiedono di calcolare la **somma** e la **somma della somma**, due valori che serviranno per fare i controlli lato ricevente.

Facciamo l'esempio con un messaggio di 4 caratteri. Vogliamo spedire il messaggio wxyz. Calcoliamo somma e somma della somma.

La **somma** si calcola come valore della cifra corrente più la somma precedente. La **somma della somma** si calcola come somma corrente più la somma della somma precedente.

Otteniamo quindi

$$\begin{bmatrix} C & \text{somma} & \text{somma della somma} \\ w & w & w \\ x & w+x & 2w+x \\ y & w+x+y & 3w+2x+y \\ z & w+x+y+z & 4w+3x+2y+z \end{bmatrix}$$

Il **carattere di controllo** aggiunto alla fine deve far si che la sua somma di somma sia congrua a 0 modulo numero dei caratteri.

Esempio 2.5.1: Supponiamo di avere 37 lettere che codifico con:

- $[0, 9]$ per i numeri da 0 a 9:
- $[10, 35]$ per le lettere maiuscole;
- $[36]$ per lo spazio.

Il messaggio da spedire sul canale è «3b 8». La tabella è:

$$\begin{bmatrix} C & \text{valore} & \text{somma} & \text{somma della somma} \\ 3 & 3 & 3 & 3 \\ B & 11 & 14 & 17 \\ & 36 & 50 & 67 \\ 8 & 8 & 58 & 125 \\ \text{check} & ? & 58+? & 183+? \end{bmatrix}$$

Il numero da aggiungere è 2, codificato con 2, quindi la stringa che spediamo sul canale è «3b 82» con l'ultimo carattere che è di controllo.

Cosa succede se mi arriva la stringa «3b82»? Calcolo:

$$\begin{bmatrix} C & \text{valore} & \text{somma} & \text{somma della somma} \\ 3 & 3 & 3 & 3 \\ B & 11 & 14 & 17 \\ 8 & 8 & 22 & 39 \\ 2 & 2 & 24 & 63 \end{bmatrix}$$

Il carattere di controllo dovrebbe essere B perché $63 + 11 \equiv 0 \bmod 37$, ma noi abbiamo ottenuto 2 quindi richiederemo la stringa.

2.6. Codice ISBN

Il **codice ISBN** (*International Standard Book Nigga*) è un codice di controllo usato per identificare in maniera univoca un libro. Di solito ha due formati, a 10 o 13 cifre (*a volte entrambi assieme*).

Consideriamo il codice a 10 cifre, che usa le somme pesate. Esso è definito da:

- prima cifra che identifica il **country ID**;
- seconda cifra e terza cifra che identificano il **publisher ID**;
- cifre dalla quarta alla nona che identificano **book number**;
- decima cifra che è la cifra di controllo, il **check digit**.

L'alfabeto usato contiene 11 caratteri (*perché* 11 *è primo e va bene con i moduli*): i numeri da 0 a 9 sono codificati con loro stessi, mentre il decimo carattere è la X.

2.7. Codice UPC

Il **codice UPC** (*Universal Product Code*) è il comunissimo **codice a barre**. Anche lui è un codice pesato. Ogni codice UPC è diviso in 3 blocchi:

- **manifacturer ID** (*chi ha prodotto l'oggetto*) di 6 cifre;
- **numero dell'oggetto** di 5 cifre;
- **check digit**.

Indicizzando i caratteri da 1, il check digit si calcola come

$$3 \sum_{i \text{ dispari}} x_i + \sum_{i \text{ pari}} x_i \bmod 10.$$

Quando ricevo un codice a barre, controllo se questa somma pesata modulo 10 è uguale a 0.

3. Codici a ripetizione

I codici a ripetizioni sono molto semplici, noi vedremo il **codice a ripetizione tripla**.

Come funziona: mando dei bit di informazione, ma ognuno di questi viene triplicato in un bit di informazione e in due bit di controllo. Quindi, se $x = x_1$ allora ho

$$\mathrm{CR}(x) = x_1x_2x_3 = \begin{cases} x_2 = x_3 = 0 & \text{se } x_1 = 0 \\ x_2 = x_3 = 1 & \text{se } x_1 = 1 \end{cases}.$$

Come riconosco gli errori:

- il dato è OK quando $x_1 + x_2 = 0$ e $x_1 + x_3 = 0$;
- il dato non è OK quando cade almeno una delle due condizioni precedenti.

Siamo in presenza di rumore bianco, quindi abbiamo una probabilità p di commettere un errore, mentre abbiamo probabilità $(1-p)$ di non commettere errori. Sia $p \ll (1-p)$, ovvero faccio pochissimi errori.

Esempio 3.1: Usando un codice a ripetizione tripla spedisco $x = 000$. Noi dall'altra parte riceviamo 001. Quanto vale:

$$P(w(000) \wedge r(001)) = P(w(000))P(r(001) \mid w(000)) = \frac{p(1-p)^2}{2}.$$

Invece quanto vale:

$$P(w(111) \wedge r(001)) = P(w(111))P(r(001) \mid w(111)) = \frac{p^2(1-p)}{2}.$$

Infine, quanto vale:

$$P(r(001)) = P(r(001) \mid w(000))P(w(000)) + P(r(001) \mid w(111))p(w(111)) =$$
$$= \frac{p^2(1-p)}{2} + \frac{p(1-p)^2}{2} = p^2(1-p) + p(1-p)^2.$$

Un codice C di tipo (n, k), dove n sono i bit totali e k sono i bit di informazione, ha **code rate**

$$CR = \frac{k}{n}.$$

Questa quantità definisce la percentuale di bit di informazione che sono utilizzati rispetto al totale dei bit che definiscono il codice. Rappresenta in poche parole la **bontà** del nostro codice.

In generale, i codici a ripetizione fanno schifo come code rate. Nel nostro caso, il codice a ripetizione tripla è di tipo $(3, 1)$ e quindi il code rate è

$$CR = \frac{1}{3}.$$

Vediamo infine come avvengono i calcoli per questo codice a ripetizione tripla.

Definiamo G la **matrice generatrice** e H la **matrice di parità** del nostro codice.

Nel nostro caso, la **matrice generatrice** G è la matrice che moltiplica s nella seguente formula:

$$(x_1x_2x_3) = s(111).$$

Il valore s indica il carattere che sto mandando, mentre le x_i sono le possibili parole del codice.

La **matrice di parità** H invece è

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}.$$

Questa rappresenta il sistema di equazioni lineare omogeneo che mi definisce il codice, ovvero i due check che abbiamo messo sopra.

Quando riceviamo una parola $x = (x_1 x_2 x_3)$ dobbiamo verificare che

$$Hx^T = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Questo codice è molto carino, ma se c'è un doppio errore non lo riusciamo a riconoscere.

4. Codice di Hamming

4.1. Definizione

Il buon **Richard Hamming** lavorava ai Bell Labs negli anni '40. Come detto nell'introduzione, si era rotto i maroni di aspettare il lunedì per vedere il fallimento dei suoi job mandati in esecuzione sulle macchine che aveva in laboratorio, quindi ha cercato di creare un codice che riconoscesse l'errore e lo sistemasse, così da evitare altri sbattimenti. Hamming era un matematico nell'Illinois, sapeva come fare tutto ciò, e infatti nel 1950 pubblica il suo lavoro.

Il **codice di Hamming** è un codice di tipo $(7, 4)$. Quello che spediamo sul canale è il vettore

$$(p_1, p_2, p_3 \mid s_1, s_2, s_3, s_4)$$

con s vettore che contiene gli s_i e che caratterizza quello che vogliamo spedire *«nudo e crudo»* sul canale mentre i p_i sono **bit di controllo**.

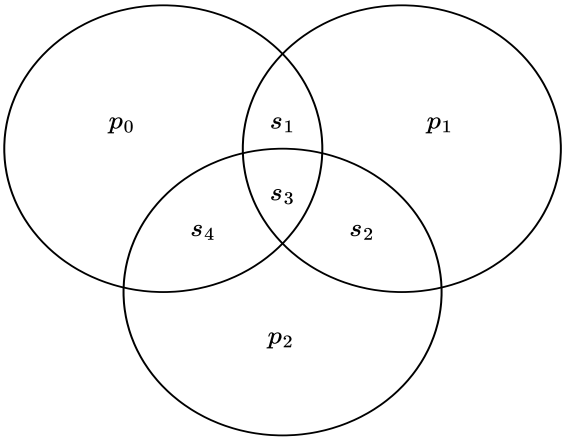
Il codice di Hamming ha come vincoli le seguenti equazioni lineari:

$$\begin{cases} p_1 = s_1 + s_3 + s_4 \\ p_2 = s_1 + s_2 + s_3. \\ p_3 = s_2 + s_3 + s_4 \end{cases}$$

Possiamo calcolare la matrice H da questi vincoli come la matrice

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

Nel 1985 un ingegnere ha proposto una spiegazione molto semplice di quello che Hamming aveva pensato. Ho a disposizione tre sfere, che sono le **sfere di influenza** dei bit di controllo. Con sfera di influenza si intende l'insieme di bit che sono coperti da un certo bit di controllo. Le tre sfere di influenza formano il **grafico di Mc-Eliece**. Vediamolo con un esempio.



Esempio 4.1.1: Abbiamo ricevuto la parola $x = 011 \mid 1100$ dal canale, è corretta?

Popoliamo il grafico di Mc-Eliece:

- per $p_0 = 0$ abbiamo $1 + 0 + 0$, quindi abbiamo un errore;
- per $p_1 = 1$ abbiamo $1 + 1 + 0$, quindi abbiamo un errore;
- per $p_2 = 1$ abbiamo $1 + 0 + 0$, quindi questo va bene.

Prendiamo i due insiemi che sono errati, quindi p_0 e p_1. Di questi insiemi prendiamo i bit che stanno nelle intersezioni, quindi s_1 e s_3. Dobbiamo capire quale dei due correggere:
- se s_3 cambia sistemiamo i due insiemi ma rendiamo sbagliato l'insieme p_2: infatti, s_3 è già corretto come bit;
- visto questo, dobbiamo sistemare s_1.

Esempio 4.1.2: Abbiamo ricevuto la parola $x = 111 \mid 0100$ dal canale, è corretta?

Popoliamo il grafico di Mc-Eliece:
- per $p_0 = 1$ abbiamo $0 + 0 + 0$, quindi abbiamo un errore;
- per $p_1 = 1$ abbiamo $0 + 1 + 0$, quindi questo va bene;
- per $p_2 = 1$ abbiamo $1 + 0 + 0$, quindi questo va bene.

In questo caso abbiamo un solo insieme errato. Quando succede questo, dobbiamo modificare il bit di controllo: infatti, se modificassimo uno dei tre bit coperti da p_0 renderemmo sbagliati gli altri insiemi.

Come vediamo, il codice di Hamming è molto potente perché riesce a rilevare e correggere anche i propri bit di controllo. Purtroppo, questo codice funziona se assumiamo di avere al più un errore.

Infine, vediamo una bella proprietà del codice di Hamming. Dato x letto dal canale con un errore, se calcoliamo Hx otteniamo una colonna di H che mi indica quale era il bit errato.

Infatti, se definiamo $x' = x + e$, con e vettore con un 1 nella posizione dell'errore, e calcoliamo

$$H(x')^T = Hx^T + He^T = 0^T + He^T$$

otteniamo la colonna che identifica l'errore.

4.2. Distanza di Hamming

La **distanza di Hamming** tra due parole di codice x_0 e x_1 è il numero di bit nel quale differiscono nelle stesse posizioni.

Usiamo questa definizione della distanza di Hamming perché la distanza euclidea in $\mathbb{Z}_2$ non funziona: infatti, notiamo che

$$00, 11 \implies \sqrt{(0-1)^2 + (0-1)^2} = \sqrt{2} = 0 \quad (????) \quad .$$

Useremo la distanza di Hamming per capire quanti errori un codice può rilevare e correggere.

Teorema 4.2.1 (*Non so se è un teorema*): Sia C un codice correttore di tipo (n, k) e sia d la minima distanza di Hamming che esiste tra le parole di C.

Allora C rileva

$$d-1$$

errori e ne corregge

$$t = \left\lfloor \frac{d-1}{2} \right\rfloor.$$

Possiamo vedere queste due quantità come due bolle su un asse cartesiano: su di esso poniamo le due parole di codice x_0, x_1 e le due bolle definite dalle due quantità precedenti. Non ho voglia di disegnare, mi dispiace tanto.

Non posso andare oltre x_0 (*partendo dall'altra parola*) perché x_0 è una parola di codice.

4.3. Rilevazione e correzione errori

La distanza di Hamming è necessaria nella definizione del numero di errori che possono essere rilevati e corretti da un codice.

Un codice P **rileva** z errori se

$$\forall x \in P \quad \forall x' \in M_n/P \Longrightarrow 0 < d(x,x') \leq z.$$

Un codice P **corregge** t errori se

$$\forall x,y \in P \mid x \neq y \quad \wedge \quad \forall x' \in M_n/P \Longrightarrow d(x,x') \leq t \wedge d(x',y) > t.$$

Teorema 4.3.1: Condizione necessaria e sufficiente affinché il codice P rilevi z errori è che

$$d(P) \geq z+1.$$

Teorema 4.3.2: Condizione necessaria e sufficiente affinché il codice P corregga t errori è che

$$d(P) \geq 2t+1.$$

In questi due teoremi, $d(P)$ indica la **distanza di Hamming minima** tra due parole del codice.

Come facciamo a sapere il valore $d(P)$?

Non so come, ma il valore $\omega(P)$, che indica il minimo numero di 1 presenti nelle parole di codice, è in realtà uguale a $d(P)$, quindi si gode.

5. Codici lineari

Sia C un codice (n, k) che mappa k bit di informazione della parola s in n bit della parola x. Diciamo che C è un **codice lineare** se esiste una **matrice generatrice** G di dimensione $k \times n$ e una **matrice di parità** H di dimensione $n - k \times n$ tale che

$$(x_1 \ \dots \ x_n) = (s_1 \ \dots \ s_k)G$$

e il controllo di appartenenza sulle parole del codice avviene con

$$H(x_1 \ \dots \ x_n)^T = \underline{0}^T.$$

Il codice di Hamming e il codice a ripetizione tripla sono codici lineari.

5.1. Definizione rigorosa

Dato A un campo finito, definiamo M_n lo **spazio dei messaggi** di ordine n su A, ovvero lo spazio lineare

$$M_n = \{x = (x_1, ..., x_n) \mid x_i \in A\}$$

nel quale valgono due regole:

1. presi $x, x' \in M_n$ allora $x + x' = (x_1 + x'_1, ..., x_n + x'_n) \in M_n$;
2. preso $x \in M_n$ e preso $s \in A$ allora $s \cdot x = (s \cdot x_1, ..., s \cdot x_n) \in M_n$.

Lo spazio M_n ha dimensione n e ha cardinalità 2^n (*siamo in binario*).

Un **codice** P è un sottospazio di M_n di dimensione k e di cardinalità 2^k (*siamo sempre in binario*).

Fissata una base

$$\mathcal{B} = \{e_1, ..., e_k\}$$

per il codice P, per definizione di base ogni elemento x di P potrà essere scritto come combinazione lineare dei vettori della base, ovvero

$$\exists! u \in \mathbb{Z}_2^k \mid x = u \begin{pmatrix} e_1 \\ \vdots \\ e_k \end{pmatrix}.$$

Il vettore formato dagli elementi della base $\mathcal{B}$ lo possiamo vedere come la **matrice generatrice** che abbiamo visto a inizio capitolo. Quello che abbiamo definito ora è un codice di tipo (n, k).

Un **codice lineare** è definito come l'insieme delle soluzioni $x = (x_1, ..., x_n)$ del sistema lineare di $n - k$ equazioni e n incognite

$$Hx = \underline{0}.$$

Dato $x \in P$ spedito sul canale e dato $y \in M_n$ valore ricevuto, chiamiamo **scheda d'errore** e il polinomio/vettore differenza tra quello ricevuto e quello inviato, ovvero

$$e = y - x.$$

Il **resto** della divisione tra la parola ricevuta $y(x)$ e il generatore $g(x)$ è detto **sindrome**, e si indica con $r(x)$. Sembra un concetto nuovo, ma l'abbiamo già visto, senza nome, nel codice di Hamming: se $Hx^T \neq \underline{0}^T$ allora il risultato è una colonna di H che indica il bit errato. Questa colonna è esattamente la sindrome che abbiamo appena definito.

Abbiamo citato più volte H ma non l'abbiamo ancora definita. Facciamolo allora.

Sia H la **matrice di parità** di P, formata dai coefficienti del sistema lineare $(n-k) \times n$ che definisce il codice P, ovvero l'insieme di regole che devono essere verificate per essere dentro P. Notiamo come la matrice H non sia unica, perché posso scambiare tra loro le righe, ma anche sommarle e tanto altro, ma sempre nei limiti definiti dal buon vecchio Gauss.

Abbiamo citato anche G, ma ancora non l'abbiamo definita pienamente. Faccio anche questo ora.

Definiamo G **matrice generatrice** di P come la matrice di dimensione $k \times n$ le cui righe sono i k vettori di una base $\mathcal{B}$ di P. Come abbiamo detto prima, allora

$$\forall x \in P \quad \exists! u \in \mathbb{Z}_2^k \mid x = uG.$$

Una matrice G è in **forma canonica** se è nella forma

$$G_{k \times n} = \left[I_{k \times k} \mid D_{k \times (n-k)} \right].$$

Ogni matrice G può essere messa in forma canonica usando Gauss o metodi simili.

Esiste una sorte di relazione tra le matrici G e H? Ebbene si.

Data G in forma canonica, la matrice H associata è

$$H_{(n-k) \times n} = \left[-D^T_{(n-k) \times k} \mid I_{(n-k) \times (n-k)} \right].$$

6. Codici ciclici

Un **codice ciclico** è un codice lineare che è dotato dell'operazione di **shift ciclico verso destra**.

Un codice lineare P si dice ciclico se

$$\forall x = (x_1, ..., x_n) \quad x' = x \gg 1 = (x_n, x_1, ..., x_{n-1}) \in P.$$

Molto comodo lo shift verso destra perché nei campi di Galois questa operazione si tramuta in una semplice moltiplicazione per x con una operazione di modulo per il polinomio irriducibile.

Definiamo il **grado** del polinomio $p(x)$ il numero

$$\text{gr}[p(x)] = \max(i \mid p[i] = 1).$$

Teorema 6.1: Sia P un codice ciclico di ordine n e sia $a(x) \in P$. Vale

$$\forall p(x) \in P \mid \text{gr}[p(x)a(x)] < n \implies p(x)a(x) \in P.$$

Vediamo come è fatta la **matrice generatrice** G di un codice ciclico.

Teorema 6.2: Sia P un codice ciclico di tipo (n, k), allora:

1. P contiene sempre almeno una parola $g(x)$ di grado $n - k$ tale che tutte le parole del codice P sono multiple di $g(x)$; la parola $g(x)$ è il nostro **polinomio generatore** e ricopre il ruolo di $a(x)$ nel teorema precedente;
2. trovata la parola $g(x)$, la **matrice generatrice** G è tale che

$$G_{k \times n} = \begin{bmatrix} g(x) \\ x \cdot g(x) \\ x^2 \cdot g(x) \\ ... \\ x^{k-1} \cdot g(x) \end{bmatrix},$$

ovvero ogni riga (*tranne la prima*) viene ottenuta dalla precedente tramite uno shift ciclico.

Teorema 6.3: Sia $g(x)$ il polinomio generatore di un codice ciclico P di tipo (n, k), allora $g(x)$ è divisore proprio di $x^n - 1$.

Il polinomio generatore $g(x)$ **non è unico**: questo ci permette di definire un altro teorema.

Teorema 6.4: Ogni divisore proprio di grado

$$r \in \{1, n - k\}$$

genera un codice ciclico diverso di tipo $(n, n - r)$.

Il **polinomio di parità** di un codice ciclico P di ordine n, generato da $g(x)$, che chiamiamo $\pi(x)$, è il polinomio quoziente

$$\pi(x) = \frac{x^n - 1}{g(x)}.$$

Vediamo infine come è definita la **matrice di parità** H di un codice P.

Teorema 6.5: Sia P un codice ciclico di tipo (n, k) e sia $\pi(x)$ il polinomio di parità

$$\pi(x) = \pi_0 + \pi_1 x + ... + \underbrace{\pi_k}_{=1} x^k.$$

Allora la **matrice di parità** H è tale che

$$H_{(n-k)\times n} = \begin{bmatrix} x^{n-k-1} \cdot \pi'(x) \mid 0...0 \\ x^{n-k-2} \cdot \pi'(x) \mid 0...0 \\ ... \\ x \cdot \pi'(x) \mid 0...0 \\ \pi'(x) \mid 0...0 \end{bmatrix},$$

con

$$\pi'(x) = \underbrace{\pi_k}_{=1} + \pi_{k-1} x + ... + \pi_0 x^k$$

polinomio ottenuto invertendo i coefficienti.

Questi codici sono estremamente potenti, ma hanno un piccolo svantaggio: una volta definito il tipo del codice abbiamo definito di conseguenza anche tutto il resto, perché abbiamo già $\omega(P)$, che corrisponde alla distanza $d(P)$, e quindi sappiamo già quanti errori possiamo rilevare e quanti ne possiamo correggere.

I **codici BCH** fanno il contrario: decidono quanti errori vogliono correggere e costruiscono di conseguenza il codice.