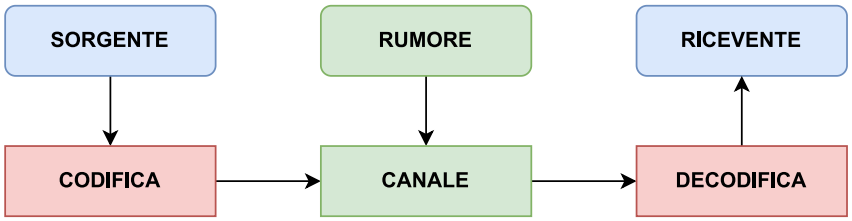
Teoria dell'Informazione e della Trasmissione

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

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

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

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

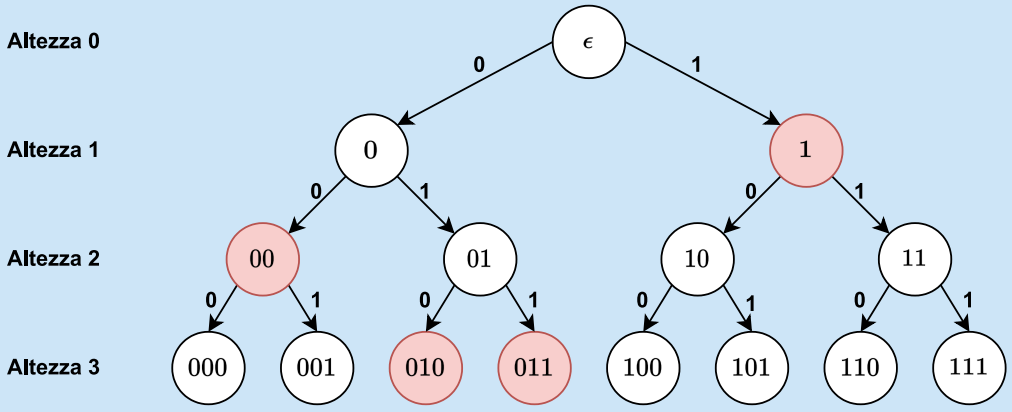
se e solo se

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

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 $l_1, ..., l_m$ tali che

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

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

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

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

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

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**.

SISTEMA DA QUA

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 $\log_d(a)$.

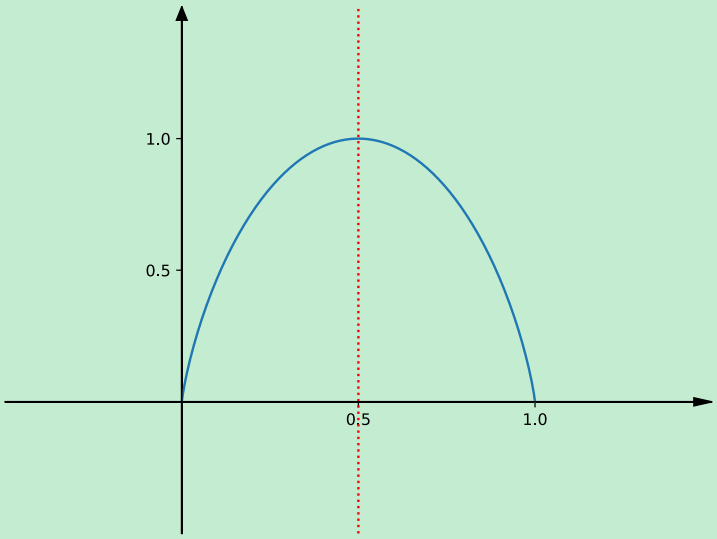
Esempio 3.2.1 (*Entropia binaria*): Sia $\mathbb{X}$ una variabile aleatoria bernoulliana tale che

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

Calcoliamo l'entropia

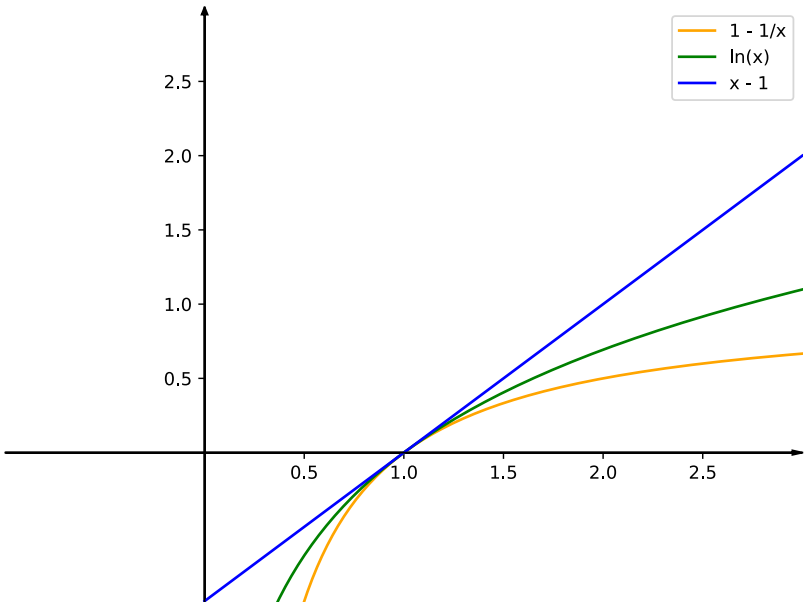
$$H_2(\mathbb{X}) = p \log_2\left(\frac{1}{p}\right) + (1-p)\log_2\left(\frac{1}{1-p}\right)$$

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.

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.

Vediamo infine 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

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

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:

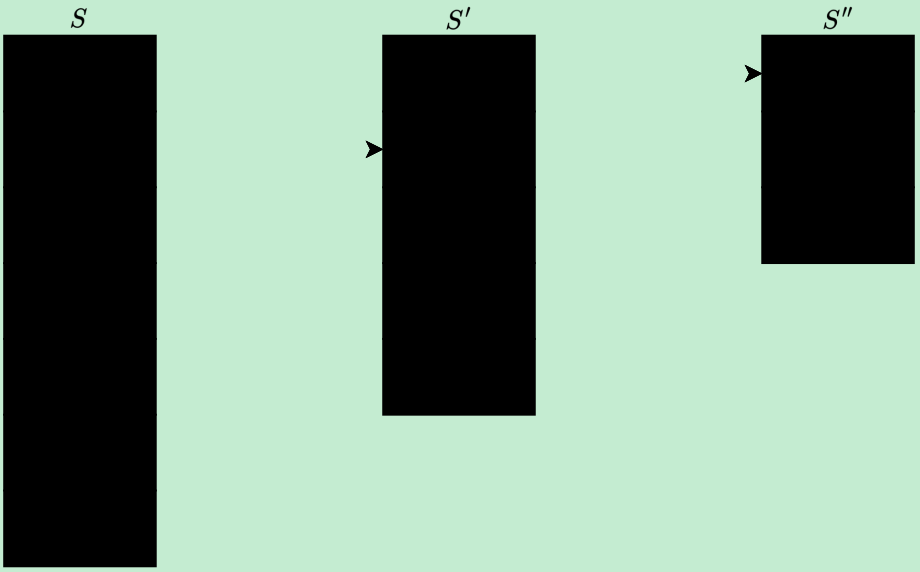
1. ordina le probabilità p_i in maniera decrescente;
2. rimuovi i d simboli meno probabili da X e crea una nuova sorgente aggiungendo un nuovo simbolo che abbia come probabilità la somma delle probabilità dei simboli rimossi;
3. se la nuova sorgente ha più di d simboli torna al punto 1.

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

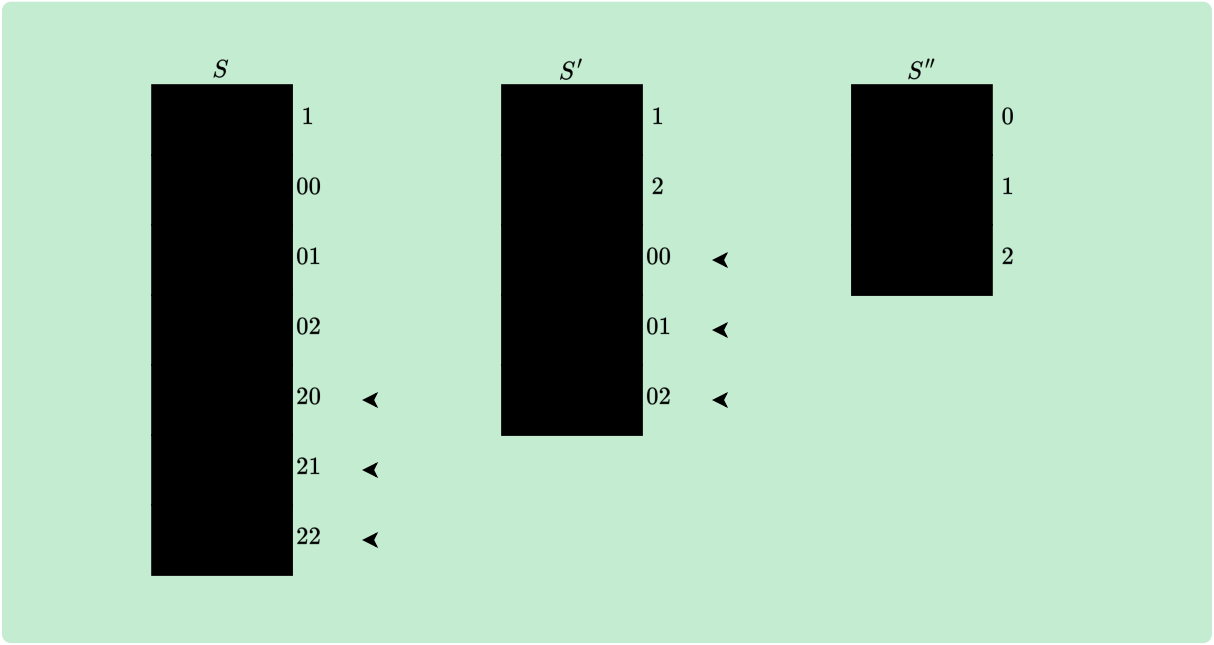
- $S = \{s_1, ..., s_7\}$ e
- $P = \{0.3, 0.2, 0.2, 0.1, 0.1, 0.06, 0.04\}$,

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

$$\begin{aligned} \mathbb{E}\left[l_{\overline{c}_2}\right] &= \sum_{x \in X' \mid x \neq z} p'(x)l_{\overline{c}_2}(x) + p(u)\Big(l_{c_2'}(z) + 1\Big) + p(v)\Big(l_{c_2'}(z) + 1\Big) = \\ &= \sum_{x \in X' \mid x \neq z} p'(x)l_{\overline{c}_2}(x) + p'(z)l_{c_2'}(z) + p'(z) = \\ &= \mathbb{E}\left[l_{c_2'}\right] + p'(z) \\ &\geq \mathbb{E}[l_{c'}] + p'(z). \end{aligned}$$

Mettendo insiemi i due risultati otteniamo

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

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

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