

# Teoria dell'informazione e della trasmissione

## Indice

<b>1. Lezione 01</b>	<b>3</b>
1.1. Introduzione	3
1.1.1. Storia	3
1.1.2. Shannon vs Kolmogorov	3
1.1.3. Obiettivi di Shannon	3
1.1.4. Primo teorema di Shannon	4
1.1.5. Secondo teorema di Shannon	5
1.2. Codifica della sorgente	6
1.2.1. Introduzione matematica	6
1.2.2. Prima applicazione	6
1.2.3. Modello statistico	7
<b>2. Lezione 02</b>	<b>8</b>
2.1. Codici	8
2.1.1. Codice univocamente decodificabile	8
2.1.2. Codice istantaneo	8
2.1.3. Gerarchia dei codici	9
2.2. Disuguaglianza di Kraft	10
2.2.1. Definizione	10
2.2.2. Applicazione	11
<b>3. Lezione 03</b>	<b>13</b>
3.1. Codice di Shannon	13
3.1.1. Definizione	13
3.1.2. Esempi	14
3.1.2.1. Base / migliore	14
3.1.2.2. Degenere	14
3.1.2.3. Equiprobabile	14
3.1.3. Entropia	15
3.2. Codice di Huffman	15
3.2.1. Definizione	16
<b>4. Lezione 04</b>	<b>19</b>
4.1. Entropia	19
4.1.1. Introduzione	19
4.1.2. Cambio di base	19
4.1.3. Esempio	19
4.1.4. Proprietà	20
4.2. Sardinias-Patterson	23
4.2.1. Definizione	23
4.2.2. Esempi	24
<b>5. Lezione 05</b>	<b>25</b>
5.1. Upper bound valore atteso	25
5.2. Estrazione a blocchi	25

5.3. Primo teorema di Shannon .....	26
5.4. Approssimare i modelli sorgente .....	27
<b>6. Lezione 06 .....</b>	<b>28</b>
6.1. Algoritmo di Huffman .....	28
6.2. Codici di Huffman .....	28
<b>7. Lezione 07 .....</b>	<b>31</b>
7.1. Codici di Huffman .....	31
7.2. Disuguaglianza di Kraft-McMillan .....	31
<b>8. Lezione 08 .....</b>	<b>33</b>
8.1. Esercizi di probabilità .....	33
8.1.1. Esercizio 01 .....	33
8.1.2. Esercizio 02 .....	33
8.1.3. Esercizio 03 .....	33
8.1.4. Esercizio 04 .....	33
8.2. Numero di bit necessari .....	33
8.3. Esercizi su entropia .....	34
8.3.1. Esercizio 01 .....	34
8.3.2. Esercizio 02 .....	34
8.3.3. Esercizio 03 .....	34
8.3.4. Esercizio 04 .....	34
8.3.5. Esercizio 05 .....	34

# 1. Lezione 01

## 1.1. Introduzione

### 1.1.1. Storia

La **teoria dell'informazione** nasce nel 1948 grazie a **Claude Shannon** (1916-2001), un impiegato alla "Telecom" americana al quale sono stati commissionati due lavori: data una comunicazione su filo di rame, si voleva sfruttare tutta la capacità del canale, ma al tempo stesso correggere gli errori di trasmissione dovuti al rumore presente.

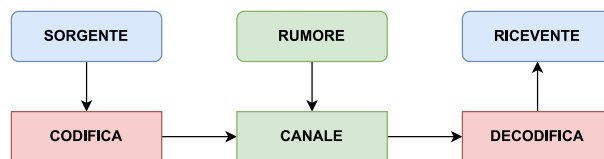
Nel luglio 1948 infatti viene pubblicato l'articolo "*A Mathematical Theory of Communication*" da parte di Bell Labs, dove Shannon pone le basi della teoria dell'informazione.

Ma non è l'unico personaggio che lavora in questo ambito: infatti, nel 1965, tre matematici russi pubblicano degli articoli che vanno a perfezionare il lavoro fatto anni prima da Shannon.

I tre matematici sono Gregory Chaitin (1947-), Ray Solomonoff (1926-2009) e **Andrey Kolmogorov** (1903-1987), ma si considerano solo gli articoli di quest'ultimo poiché ai tempi era molto più famoso dei primi due.

### 1.1.2. Shannon vs Kolmogorov

La situazione generica che troveremo in quasi la totalità dei nostri studi si può ridurre alla comunicazione tra due entità tramite un **canale affetto da rumore**.



Quello che distingue Shannon da Kolmogorov è l'approccio: il primo propone un **approccio ingegneristico**, ovvero tramite un modello formato da una distribuzione di probabilità si va a definire cosa fa *in media* la sorgente, mentre il secondo propone un **approccio rigoroso e formale**, dove sparisce la nozione di media e si introduce la nozione di sorgente in modo *puntuale*.

In poche parole, dato un messaggio da comprimere:

- Shannon direbbe "lo comprimo *in media* così, e lo comprimerei così anche se il messaggio fosse totalmente diverso";
- Kolmogorov direbbe "lo comprimo *esattamente* così, ma lo comprimerei in modo totalmente diverso se il messaggio fosse diverso".

### 1.1.3. Obiettivi di Shannon

Gli obiettivi che Shannon vuole perseguire sono due:

- **massimizzare** l'informazione trasmessa *ad ogni utilizzo del canale*;
- **minimizzare** il numero di errori di trasmissione dovuti alla presenza del rumore nel canale.

La parte "*ad ogni utilizzo del canale*" viene inserita per dire che, ogni volta che si accede al canale, deve essere utilizzato tutto, mentre senza questa parte una sorgente potrebbe mandare l'1% del messaggio ad ogni accesso al canale, mandandolo sì tutto ma senza sfruttare a pieno la banda.

Shannon risolverà questi due problemi con due importantissimi teoremi:

- **I° teorema di Shannon**, che riguarda la *source coding*, ovvero la ricerca di un codice per rappresentare i messaggi della sorgente che massimizzi l'informazione spedita sul canale, ovvero massimizzi la sua **compressione**;
- **II° teorema di Shannon**, che riguarda la *channel coding*, ovvero la ricerca di un codice per rappresentare i messaggi della sorgente che minimizzi gli errori di trasmissione dovuti alla presenza del rumore nel canale.

L'approccio che viene usato è quello *divide-et-impera*, che in questo caso riesce a funzionare bene e riesce ad unire i risultati dei due teoremi di Shannon grazie al **teorema di codifica congiunta sorgente-canale** e ad alcune relazioni che legano i due problemi descritti.

In un caso generale del *divide-et-impera* si ricade in una soluzione sub-ottimale.

#### 1.1.4. Primo teorema di Shannon

Il primo problema da risolvere è il seguente: come è distribuita l'informazione all'interno di un documento?

Vediamo due esempi dove un documento viene spedito su un canale e alcune informazioni vengono perse per colpa del rumore presente nel canale.



In questo primo esempio notiamo che, nonostante l'informazione persa sia sostanziosa, possiamo in qualche modo “risalire” a quello perso per via delle informazioni che troviamo “nelle vicinanze”.



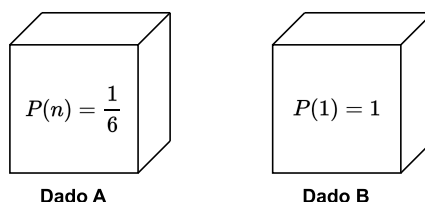
In questo secondo esempio notiamo invece che, nonostante l'informazione persa sia molto meno rispetto a quella precedente, “risalire” al contenuto perso è molto più difficile.

Questi due esempi dimostrano come l'informazione contenuta in un documento **non** è uniforme, e quindi che una distorsione maggiore non implica una perdita maggiore di informazioni.

L'obiettivo del primo teorema di Shannon è eliminare le informazioni inutili e ridondanti, comprimendo il messaggio per poter utilizzare il canale per inviare altre informazioni.

Quello che facciamo è concentrare l'informazione, rendendola **equamente distribuita**, quindi impossibile da ridurre ancora e contenente solo informazioni importanti.

Vediamo un altro esempio: supponiamo di avere due dadi a sei facce, uno *normale* e uno *truccato*, e supponiamo di tirarli assieme per un numero di volte molto grande. Quale dei due dadi mi dà più informazioni?



La risposta è quello *normale*: nel lungo il dado *normale* si “normalizzerà” su una probabilità di circa  $\frac{1}{6}$  per ogni possibile faccia, quindi è una sorta di evento **regolare** che mi dà tanta informazione, mentre quello truccato posso sapere già cosa produrrà, quindi è una sorta di evento **prevedibile** che mi dà poca informazione.

In poche parole, massimizzo la probabilità di “ottenere” informazioni se le probabilità di estrarre un simbolo sono uguali.

### 1.1.5. Secondo teorema di Shannon

Il secondo teorema di Shannon è quello più rognoso, perché si occupa della *channel coding*, ovvero di una codifica che permetta di minimizzare l’informazione persa durante la trasmissione.

Vogliamo questo perché l’informazione che passa sul canale è compressa, quindi qualsiasi bit perso ci fa perdere molte informazioni, non essendoci ridondanza.

Quello che viene fatto quindi è aggiungere **ridondanza**, ovvero più copie delle informazioni da spedire così che, anche perdendo un bit di informazione, lo si possa recuperare usando una delle copie inviate.

La ridondanza che aggiungiamo però è **controllata**, ovvero in base al livello di distorsione del canale utilizzato si inviano un certo numero di copie.

In un **canale ideale** la ridondanza è pari a 0, mentre per canali con rumore viene usata una matrice stocastica, che rappresenta la distribuzione probabilistica degli errori.

IN/OUT	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>
<i>a</i>	0.7	0.0	0.1	0.1	0.1
<i>b</i>	0.2	0.8	0.0	0.0	0.0
<i>c</i>	0.1	0.0	0.6	0.2	0.1
<i>d</i>	0.0	0.0	0.2	0.5	0.3
<i>e</i>	0.0	0.0	0.0	0.0	1.0

Ogni riga *i* rappresenta una distribuzione di probabilità che definisce la probabilità che, spedito il carattere *i*, si ottenga uno dei valori *j* presenti nelle colonne. Ovviamente, la somma dei valori su ogni riga è uguale a 1.

Se il canale è ideale la matrice risultante è la matrice identità.

## 1.2. Codifica della sorgente

Andiamo a modellare e formalizzare il primo problema con un modello statistico, utilizzando però un approccio *semplice e bello*: assumiamo che le regole di compressione non siano dipendenti dalle proprietà di un dato linguaggio.

Ad esempio, data la lettera “H” nella lingua italiana, ho più probabilità che esca la lettera “I” piuttosto che la lettera “Z” come successiva di “H”, ma questa probabilità nel nostro modello non viene considerata.

Il codice *zip* invece prende in considerazione questo tipo di distribuzione statistica dipendente, e infatti è molto più complicato.

### 1.2.1. Introduzione matematica

Introduciamo una serie di “personaggi” che saranno utili nella nostra modellazione:

- insieme  $X$ : insieme finito di simboli che compongono i messaggi generati dalla sorgente;
- messaggio  $\bar{x}$ : sequenza di  $n$  simboli sorgente; in modo formale,

$$\bar{x} = (x_1, \dots, x_n) \in X^n, \text{ con } x_i \in X \ \forall i \in \{1, \dots, n\};$$

- base  $D$ ;
- insieme  $\{0, \dots, D-1\}$ : insieme finito dei simboli che compongono il codice scelto;
- insieme  $\{0, \dots, D-1\}^+$ : insieme di tutte le possibili parole di codice esprimibili tramite una sequenza non vuota di simboli di codice; in modo formale,

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

Un altro nome per le parole di codice è sequenze  $D$ -arie;

- funzione  $c$ : funzione (nel nostro caso *codice*) che mappa ogni simbolo  $x \in X$  in una parola di codice, ovvero una funzione del tipo

$$c : X \longrightarrow \{0, \dots, D-1\}^+.$$

Questa funzione va ad effettuare un **mapping indipendente**, ovvero tutto viene codificato assumendo che non esistano relazioni tra due o più estrazioni consecutive.

### 1.2.2. Prima applicazione

Vogliamo trasmettere sul canale i semi delle carte da poker utilizzando il codice binario.

Andiamo a definire:

- $X = \{\heartsuit, \diamondsuit, \clubsuit, \spadesuit\}$  insieme dei simboli sorgente;
- $c : X \longrightarrow \{0, 1\}^+$  funzione di codifica;
- $c(\heartsuit) = 0$ ;
- $c(\diamondsuit) = 01$ ;
- $c(\clubsuit) = 010$ ;
- $c(\spadesuit) = 10$ .

La codifica proposta è sicuramente plausibile, ma ha due punti deboli:

- *ambiguità*: se ricevo “010” come possibili traduzioni ho:
  - $\clubsuit$ ;
  - $\heartsuit\spadesuit$ ;
  - $\diamondsuit\heartsuit$ ;
- *pessima compressione*: usiamo tre simboli di codice per codificare  $\clubsuit$  e solo uno per codificare  $\heartsuit$ .

Quest'ultimo punto debole viene risolto prima introducendo  $l_c(x)$  come lunghezza della parola di codice associata ad  $x \in X$ , e poi minimizzando la media di tutte le lunghezze al variare di  $x \in X$ .

### 1.2.3. Modello statistico

Per completare il modello abbiamo bisogno di un'altra informazione: la **distribuzione di probabilità**, che definisce la probabilità con la quale i simboli sorgente sono emessi.

Definiamo quindi la sorgente come la coppia  $\langle X, p \rangle$ , dove  $p$  rappresenta la probabilità prima descritta.

Aggiungiamo qualche "protagonista" a quelli già prima introdotti:

- funzione  $P_n$ : non siamo molto interessati ai singoli simboli sorgente, ma vogliamo lavorare con i messaggi, quindi definiamo

$$P_n(\bar{x}) = P_n(x_1, \dots, x_n) = \prod_{i=1}^n p(x_i).$$

Possiamo applicare la produttoria perché Shannon assume che ci sia *indipendenza* tra più estrazioni di simboli sorgente;

- variabile aleatoria  $\mathbb{X}$ : variabile aleatoria  $\mathbb{X} : X \rightarrow \mathbb{R}$  che rappresenta un'estrazione di un simbolo sorgente;
- insieme  $\mathbb{D}$ : già definito in precedenza come  $\{0, \dots, D-1\}$ ;
- funzione  $c$ : già definita in precedenza, ma che riscriviamo come  $c : X \rightarrow \mathbb{D}^+$ .

Con questo modello, fissando  $X$  insieme dei simboli sorgente e  $D$  base, vogliamo trovare un codice  $c : X \rightarrow \mathbb{D}^+$  che realizzi la migliore compressione, ovvero che vada a minimizzare il *valore atteso* della lunghezza delle parole di codice, definito come  $\mathbb{E}[l_c] = \sum_{x \in X} l_c(x)p(x)$ .

La strategia che viene utilizzata è quella dell'alfabeto Morse: utilizziamo parole di codice corte per i simboli che sono generati spesso dalla sorgente, e parole di codice lunghe per i simboli che sono generatori raramente dalla sorgente.

Il primo problema che incontriamo è quello di evitare la *codifica banale*: se usassi il codice

$$c(x) = 0/1 \quad \forall x \in X$$

avrei sì una codifica di lunghezza minima ma sarebbe impossibile da decodificare.

Dobbiamo imporre che il codice  $c$  sia **iniettivo**, o *non-singolare*.

## 2. Lezione 02

### 2.1. Codici

Come detto nella scorsa lezione, andiamo a considerare dei codici non singolari per risolvere la problematica dei due *codici banali*, che minimizzavano sì il valore atteso delle lunghezze delle parole di codice  $l_c(x)$ , ma rendevano impossibile la decodifica.

Andiamo ora a raffinare ulteriormente i codici singolari presi in considerazione.

#### 2.1.1. Codice univocamente decodificabile

Come detto nella scorsa lezione, siamo interessati alle parole che vengono generate dalla mia sorgente, e non ai singoli caratteri.

Andiamo a definire:

- $X = \{\heartsuit, \diamondsuit, \clubsuit, \spadesuit\}$  insieme dei simboli sorgente;
- $c : X \rightarrow \{0, 1\}^+$  funzione di codifica;
- $c(\heartsuit) = 0$ ;
- $c(\diamondsuit) = 01$ ;
- $c(\clubsuit) = 010$ ;
- $c(\spadesuit) = 10$ .

La codifica  $c$  scelta è sicuramente non singolare, però abbiamo delle problematiche a livello di decodifica: infatti, possiamo scrivere 01001 come

- $c(\clubsuit, \diamondsuit)$ ;
- $c(\diamondsuit, \heartsuit, \diamondsuit)$ ;
- $c(\heartsuit, \spadesuit, \diamondsuit)$

e lato ricevente questo è un problema, perché “unendo” le singole codifiche non otteniamo una parola che è generabile in modo unico.

Introduciamo quindi l'**estensione** di un codice  $c$ : essa è un codice  $C : X^+ \rightarrow \mathbb{D}^+$  definito come  $C(x_1, \dots, x_n) = c(x_1) \dots c(x_n)$  che indica la sequenza ottenuta giustapponendo le parole di codice  $c(x_1), \dots, c(x_n)$ .

L'estensione  $C$  di un codice  $c$  non eredita in automatico la proprietà di non singolarità di  $c$ .

Un codice  $c$  è **univocamente decodificabile** quando la sua estensione  $C$  è non singolare.

Tramite l'**algoritmo di Sardinas-Patterson** siamo in grado di stabilire se un codice è univocamente decodificabile in tempo  $O(mL)$ , dove  $m = |X|$  e  $L = \sum_{x \in X} l_c(x)$

#### 2.1.2. Codice istantaneo

I codici univocamente decodificabili sono ottimali? Sto minimizzando al meglio  $\mathbb{E}[l_c]$ ?

Prima di chiederci questo vogliamo creare un codice che rispetti un'altra importante proprietà: permetterci di decodificare *subito* quello che mi arriva dal canale senza dover aspettare tutto il flusso.

Andiamo a definire:

- $X = \{\heartsuit, \diamondsuit, \clubsuit, \spadesuit\}$  insieme dei simboli sorgente;
- $c : X \rightarrow \{0, 1\}^+$  funzione di codifica;
- $c(\heartsuit) = 10$ ;
- $c(\diamondsuit) = 00$ ;
- $c(\clubsuit) = 11$ ;
- $c(\spadesuit) = 110$ .



Supponiamo di spedire sul canale la stringa 11000...00, e poniamoci lato ricevente. In base al numero di 0 inviati abbiamo due possibili decodifiche:

- #0 pari: decodifichiamo con  $\clubsuit\heartsuit\dots\heartsuit$ ;
- #0 dispari: decodifichiamo con  $\spadesuit\heartsuit\dots\heartsuit$ .

Nonostante si riesca perfettamente a decodificare, e questo è dato dal fatto che  $c$  è un codice univocamente decodificabile, dobbiamo aspettare di ricevere tutta la stringa per poterla poi decodificare, ma in ambiti come lo *streaming* questa attesa non è possibile.

Un altro problema lo riscontriamo a livello di memoria: supponiamo che lato sorgente vengano codificati dei dati dell'ordine dei terabyte, per il ricevente sarà impossibile tenere in memoria una quantità simile di dati per fare la decodifica alla fine della ricezione.

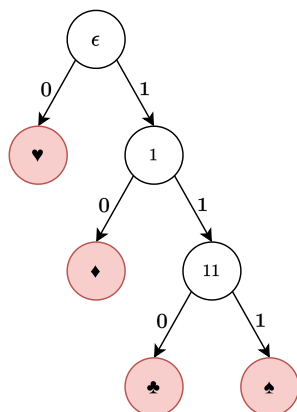
Introduciamo quindi i **codici istantanei**, particolari codici che permettono di decodificare quello che arriva dal canale, appunto, in modo *istantaneo* senza aspettare.

Un codice si dice istantaneo se nessuna parola di codice è *prefissa* di altre.

Andiamo a definire:

- $X = \{\heartsuit, \diamondsuit, \clubsuit, \spadesuit\}$  insieme dei simboli sorgente;
- $c : X \rightarrow \{0, 1\}^+$  funzione di codifica;
- $c(\heartsuit) = 0$ ;
- $c(\diamondsuit) = 10$ ;
- $c(\clubsuit) = 110$ ;
- $c(\spadesuit) = 111$ .

Il seguente codice è istantaneo, e lo notiamo osservando l'*albero* che dà origine a questo codice.



Quello che otteniamo è un albero molto sbilanciato.

### 2.1.3. Gerarchia dei codici

Cerchiamo infine di definire una **gerarchia** tra i codici analizzati fin'ora: se siamo sicuri che i codici non singolari siano sotto-insieme di tutti i codici (*banale*), e che i codici univocamente decodificabili siano sotto-insieme dei codici non singolari (*banale*), siamo sicuri che i codici istantanei siano un sotto-insieme dei codici univocamente decodificabili?

**Lemma** Se  $c$  è istantaneo allora è anche univocamente decodificabile.

### Dimostrazione

Andiamo a dimostrare che se  $c$  non è univocamente decodificabile allora non è istantaneo. Visto che  $c$  non è univocamente decodificabile (supponiamo sia almeno non singolare) esistono due messaggi distinti  $x_1, x_2 \in X^+$  tali che  $C(x_1) = C(x_2)$ .

Ci sono solo due modi per avere  $x_1$  e  $x_2$  distinti:

1. 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;
2. 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 sto dicendo che  $x_1$  deve avere  $x_2$  come prefisso (o viceversa), ma, come al punto precedente, otteniamo che  $c$  non è istantaneo.

Quindi il codice  $c$  non è istantaneo. □

Abbiamo quindi stabilito una gerarchia di questo tipo:

codici istantanei  $\subset$  codici univocamente decodificabili  $\subset$  codici non singolari  $\subset$  codici .

Una cosa che possiamo notare è come, aumentando di volta in volta le proprietà di un codice, il valore atteso  $\mathbb{E}[l_c]$  che vogliamo minimizzare peggiora, o al massimo rimane uguale: questo perché aggiungendo delle proprietà al nostro codice imponiamo dei limiti che aumentano in modo forzato la lunghezza delle nostre parole di codice.

## 2.2. Disuguaglianza di Kraft

### 2.2.1. Definizione

I codici istantanei soddisfano la **disuguaglianza di Kraft**, che ci permette, solo osservando le lunghezze delle parole di codice  $l_c(x)$ , di dire se esiste un codice istantaneo con quelle lunghezze.

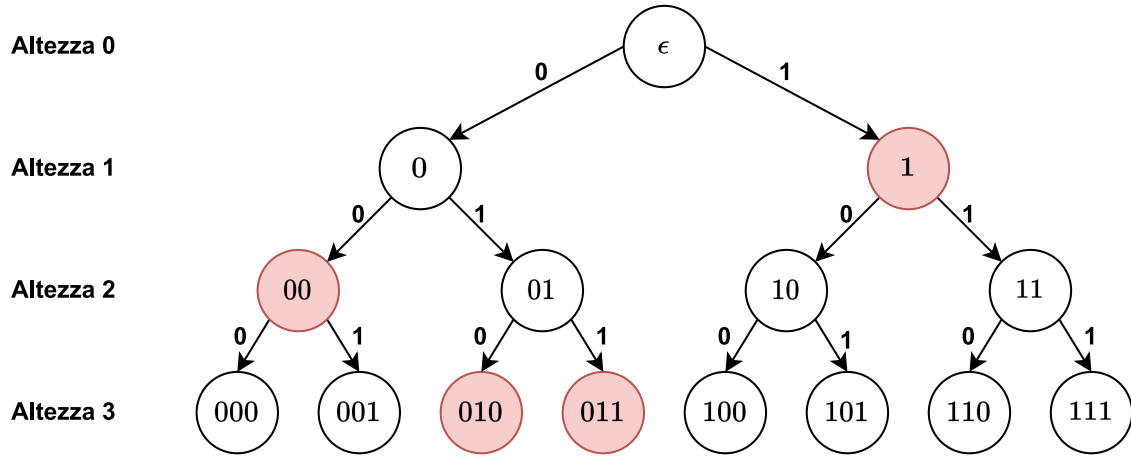
**Teorema (Disuguaglianza di Kraft)** Dati  $X = \{x_1, \dots, x_n\}$ ,  $D > 1$  e  $n$  valori interi positivi  $l_1, \dots, l_n$ , esiste un codice istantaneo  $c : X \rightarrow \mathbb{D}^+$  tale che  $l_c(x_i) = l_i$  per  $i = 1, \dots, n$  se e solo se

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

### Dimostrazione

( $\Rightarrow$ ) Sia  $l_{\max}$  la lunghezza massima delle parole di  $c$ , ovvero  $l_{\max} = \max_{i=1, \dots, n} (l_c(x_i))$ . Si consideri l'albero  $D$ -ario completo di profondità  $l_{\max}$  nel quale posizioniamo ogni parola di codice di  $c$  su un nodo dell'albero, seguendo dalla radice il cammino corrispondente ai simboli della parola. Dato che il codice è istantaneo, nessuna parola apparterrà al sotto-albero avente come radice un'altra parola di codice, altrimenti avremmo una parola di codice prefissa di un'altra. Andiamo ora a partizionare le foglie dell'albero in sottoinsiemi disgiunti  $A_1, \dots, A_n$ , dove  $A_i$  indica il sottoinsieme di foglie associate alla radice contenente la parola  $c(x_i)$ .

Nel seguente esempio consideriamo un albero binario di altezza 3, e in rosso sono evidenziate le parole di codice 1, 00, 010, e 011.



Il numero massimo di foglie di un sotto-albero di altezza  $l_i$  è  $D^{l_{\max}-l_i}$ , ma il numero massimo di foglie nell'albero è  $D^{l_{\max}}$ , quindi

$$\underbrace{\sum_{i=1}^n |A_i|}_{\text{\#foglie coperte}} = \sum_{i=1}^n D^{l_{\max}-l_i} = \sum_{i=1}^n D^{l_{\max}} \cdot D^{-l_i} = D^{l_{\max}} \sum_{i=1}^n D^{-l_i} \leq D^{l_{\max}}.$$

Dividendo per  $D^{l_{\max}}$  entrambi i membri otteniamo la disuguaglianza di Kraft.

( $\Leftarrow$ ) Assumiamo di avere  $n$  lunghezze positive  $l_1, \dots, l_n$  che soddisfano la disuguaglianza di Kraft e sia  $l_{\max} = \max_{i=1, \dots, n} (l_i)$  la profondità dell'albero  $D$ -ario ordinato e completo usato prima.

Associamo ad ogni simbolo  $x_i \in X$  la parola di codice  $c(x_i)$ , e la inseriamo al primo nodo di altezza  $l_i$  che troviamo in ordine lessicografico. Durante l'inserimento delle parole  $c(x_i)$  dobbiamo escludere tutti i nodi che appartengono a sotto-alberi con radice una parola di codice già inserita o che includono un sotto-albero con radice una parola di codice già inserita. Il codice così costruito è istantaneo, e visto che rispetta la disuguaglianza di Kraft, la moltiplichiamo da entrambi i membri per  $D^{l_{\max}}$  per ottenere

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

ovvero il numero di foglie necessarie a creare il codice non eccede il numero di foglie disponibili nell'albero.  $\square$

### 2.2.2. Applicazione

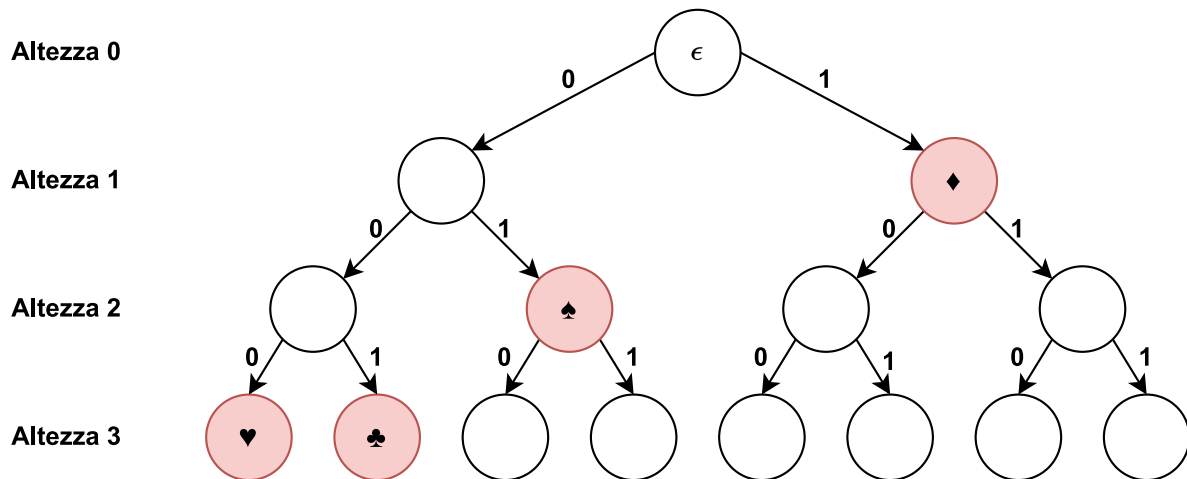
Andiamo a definire:

- $X = \{\heartsuit, \diamondsuit, \clubsuit, \spadesuit\}$  insieme dei simboli sorgente;
- $c : X \rightarrow \{0, 1\}^+$  funzione di codifica;
- $l_c(\heartsuit) = 3$ ;
- $l_c(\diamondsuit) = 1$ ;
- $l_c(\clubsuit) = 3$ ;
- $l_c(\spadesuit) = 2$ .

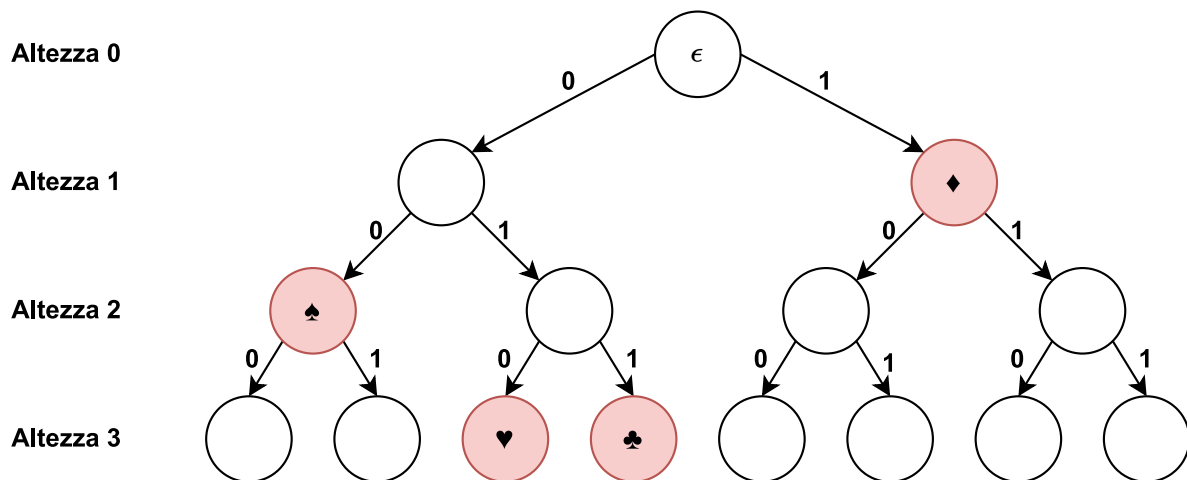
Vogliamo sapere se esiste un codice istantaneo, definito come  $c$ , aventi le lunghezze sopra definite. Controlliamo quindi se esse soddisfano la disuguaglianza di Kraft:

$$\sum_{x \in X} 2^{-l_c(x)} = 2^{-3} + 2^{-1} + 2^{-3} + 2^{-2} = \frac{1}{8} + \frac{1}{2} + \frac{1}{8} + \frac{1}{4} = 1 \leq 1.$$

Andiamo a costruire quindi un codice istantaneo con queste lunghezze costruendo il suo albero.



Il codice ottenuto è l'unico possibile?



Come vediamo, “specchiando” il sotto-albero sinistro con radice ad altezza 1 otteniamo un codice istantaneo diverso dal precedente, ma comunque possibile.

Possiamo concludere quindi che, date  $n$  lunghezze positive  $l_1, \dots, l_n$  che soddisfano la disuguaglianza di Kraft, in generale *non è unico* il codice istantaneo che si può costruire.

## 3. Lezione 03

### 3.1. Codice di Shannon

#### 3.1.1. Definizione

Abbiamo trovato un modo per verificare se un codice è istantaneo (osservare i prefissi) e un modo per verificare se  $n$  lunghezze positive possono essere usate come lunghezze di un codice istantaneo (disuguaglianza di Kraft), ma quale di questi codici istantanei è il migliore possibile?

Andiamo a vedere un modo per **costruire** un codice istantaneo a partire dai simboli sorgente e dalle loro probabilità di essere estratti dalla sorgente.

Dati il modello  $\langle X, p \rangle$  e  $D > 1$ , vogliamo trovare  $n$  lunghezze positive  $l_1, \dots, l_n$  per costruire un codice istantaneo con le lunghezze appena trovate minimizzando  $\mathbb{E}[l_c]$ , ovvero:

$$\begin{cases} \text{minimize } \sum_{i=1}^n l_i p_i \\ \text{tale che } \sum_{i=1}^n D^{-l_i} \leq 1 \end{cases}.$$

Quello che vogliamo fare è cercare  $n$  lunghezze positive  $l_1, \dots, l_n$  che minimizzino il valore atteso della lunghezza delle parole di codice e che soddisfino la disuguaglianza di Kraft.

Abbiamo a disposizione:

- $X = \{x_1, \dots, x_n\}$  insieme dei simboli sorgente, con  $|X| = n$ ;
- $P = \{p_1, \dots, p_n\}$  insieme delle probabilità, con  $p_i = p(x_i)$  e  $\sum_{i=1}^n p_i = 1$ .

Vogliamo trovare  $L = \{l_1, \dots, l_n\}$  insieme delle lunghezze delle parole di codice.

Andiamo ad unire la disuguaglianza di Kraft con la definizione di probabilità: infatti, sapendo che  $\sum_{i=1}^n p_i = 1$ , andiamo a sostituire questa sommatoria al posto del valore 1 all'interno della disuguaglianza di Kraft, ottenendo

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

Sicuramente questa disuguaglianza vale se imponiamo

$$D^{-l_i} \leq p_i \quad \forall i = 1, \dots, n$$

ma allora

$$D^{l_i} \cdot D^{-l_i} \leq p_i \cdot D^{l_i} \implies D^{l_i} \geq \frac{1}{p_i} \implies l_i \geq \log_D \frac{1}{p_i}.$$

Non sempre però il logaritmo mi rappresenta delle quantità intere, quindi andiamo ad arrotondare per eccesso questo conto:

$$l_i = \left\lceil \log_D \frac{1}{p_i} \right\rceil.$$

Abbiamo così trovato le lunghezze del mio codice istantaneo, legate in modo stretto alla probabilità di estrarre un simbolo.

Il codice così trovato viene detto **codice di Shannon**, o *codice di Shannon-Fano*, e siamo sicuri che è un codice “che fa bene”: infatti, usa tante parole di codice per simboli che vengono estratti raramente, e poche parole di codice per simboli che vengono estratti frequentemente.

Questa proprietà viene dal fatto che il logaritmo, essendo una funzione monotona crescente, produce:

- valori “grandi” quando viene calcolato con numeri “grandi”, quindi quando  $\frac{1}{p_i}$  è “grande” e quindi  $p_i$  è “piccolo”;
- valori “piccoli” quando viene calcolato con numeri “piccoli”, quindi quando  $\frac{1}{p_i}$  è “piccolo” e quindi  $p_i$  è “grande”.

### 3.1.2. Esempi

#### 3.1.2.1. Base / migliore

Supponiamo che la sorgente  $S$  emetta  $n = 4$  simboli con probabilità  $P = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}\}$ , vogliamo costruire un codice binario istantaneo.

Calcoliamo le lunghezze con l’algoritmo proposto da Shannon:

- $l_1 = \left\lceil \log_2 \frac{1}{\frac{1}{2}} \right\rceil = \lceil \log_2 2 \rceil = 1;$
- $l_2 = \left\lceil \log_2 \frac{1}{\frac{1}{4}} \right\rceil = \lceil \log_2 4 \rceil = 2;$
- $l_{3,4} = \left\lceil \log_2 \frac{1}{\frac{1}{8}} \right\rceil = \lceil \log_2 8 \rceil = 3.$

Calcoliamo il valore atteso come  $\mathbb{E}[l_c] = \sum_{i=1}^4 l_i p_i = 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{8} + 3 \cdot \frac{1}{8} = \frac{7}{4}.$

Il valore che abbiamo trovato è il migliore possibile?

La risposta è sì, e questo vale perché tutte le probabilità sono *potenze negative* della base  $D$  scelta, nel nostro caso  $D = 2$ .

Possiamo affermare questo perché le lunghezze  $l_i$  saranno esattamente uguali a  $\log_D \frac{1}{p_i}$  senza eseguire nessuna approssimazione.

#### 3.1.2.2. Degenere

Supponiamo che la sorgente  $S$  emetta  $n = 4$  simboli con probabilità  $P = \{1, 0, 0, 0\}$ , vogliamo costruire un codice binario istantaneo.

Calcoliamo le lunghezze con l’algoritmo proposto da Shannon:

- $l_1 = \left\lceil \log_2 \frac{1}{1} \right\rceil = \lceil \log_2 1 \rceil = 0;$
- $l_{2,3,4} = \left\lceil \lim_{t \rightarrow 0^+} \log_2 \frac{1}{t} \right\rceil = \lceil \lim_{t \rightarrow 0^+} \log_2 +\infty \rceil = +\infty.$

Calcoliamo il valore atteso come  $\mathbb{E}[l_c] = \sum_{i=1}^4 l_i p_i = 0 \cdot 1 + 0 \cdot (+\infty) + 0 \cdot (+\infty) + 0 \cdot (+\infty) = 0.$   
 $\underbrace{\hspace{10em}}_{0 \text{ per } t \rightarrow 0^+}$

#### 3.1.2.3. Equiprobabile

Supponiamo che la sorgente  $S$  emetta  $n = 4$  simboli con probabilità  $P = \{\frac{1}{n}, \frac{1}{n}, \frac{1}{n}, \frac{1}{n}\}$ , vogliamo costruire un codice binario istantaneo.

Calcoliamo le lunghezze con l’algoritmo proposto da Shannon:

- $l_{1,2,3,4} = \left\lceil \log_2 \frac{1}{\frac{1}{4}} \right\rceil = \lceil \log_2 4 \rceil = 2.$

Calcoliamo il valore atteso come  $\mathbb{E}[l_c] = \sum_{i=1}^4 l_i p_i = 2 \cdot \frac{1}{4} + 2 \cdot \frac{1}{4} + 2 \cdot \frac{1}{4} + 2 \cdot \frac{1}{4} = 2.$

### 3.1.3. Entropia

Nel codice di Shannon andiamo a definire  $l_i = \left\lceil \log_D \frac{1}{p_i} \right\rceil$ , quindi sostituiamo  $l_i$  nel valore atteso che stiamo cercando di minimizzare, ottenendo  $\mathbb{E}[l_c] = \sum_{i=1}^n p_i \log_D \frac{1}{p_i}$ .

La quantità che abbiamo appena scritto si chiama **entropia**, e la usiamo perché è in stretta relazione con la codifica ottimale che possiamo realizzare. In particolare, l'entropia è il *limite inferiore* alla compattezza del codice.

Tutti i valori attesi che abbiamo calcolato negli esempi precedenti sono anche le entropie delle varie sorgenti, ma che valori assume questa entropia?

Sicuramente è una quantità *positiva o nulla*, visto la somma di prodotti di fattori *positivi o nulli*.

Inoltre, raggiunge il proprio massimo quando la distribuzione di probabilità è **uniforme**, e vale

$$\sum_{i=1}^n p_i \log_D \frac{1}{p_i} = \sum_{i=1}^n \frac{1}{m} \log_D m = \mathcal{M} \cdot \frac{1}{\mathcal{M}} \cdot \log_D m = \log_D m.$$

In questo caso otteniamo un *albero perfettamente bilanciato* con le codifiche a livello delle ultime foglie.

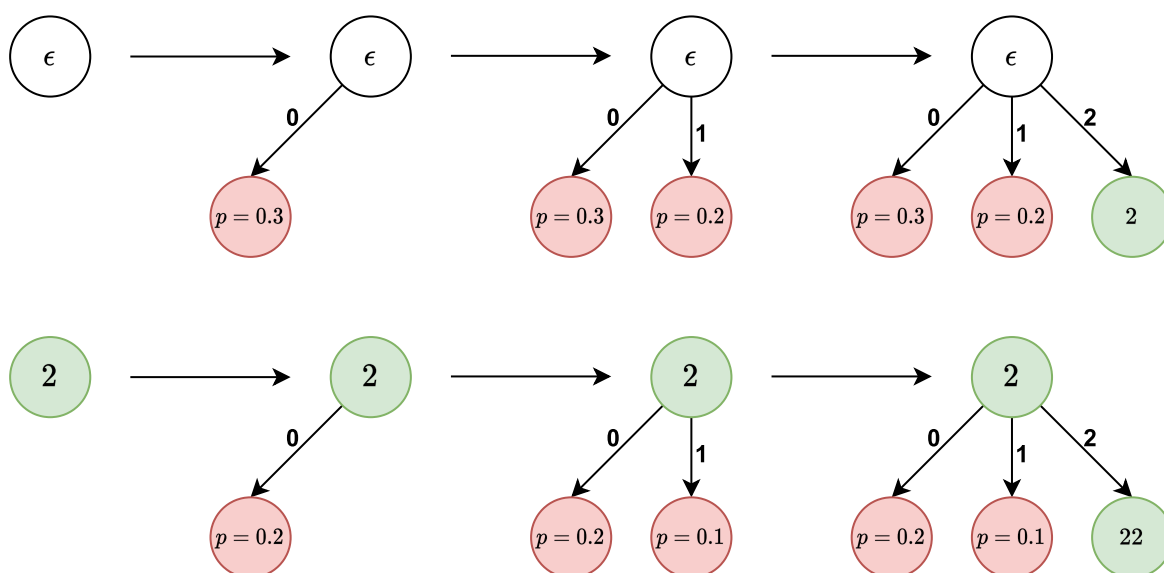
Questo mi va a dire che il codice è *compatto* e bilanciato, non spreco bit, mentre in altre situazioni ho un albero *sbilanciato* e potrei perdere dei bit.

### 3.2. Codice di Huffman

Supponiamo che la sorgente  $S$  emetta  $n = 7$  simboli con probabilità  $P = \{0.3, 0.2, 0.2, 0.1, 0.1, 0.06, 0.04\}$ . Vogliamo costruire un codice ternario istantaneo.

Qui abbiamo due diversi approcci:

- codice di Shannon: otteniamo come lunghezze 2, 2, 2, 3, 3, 3, 3;
- *grafico*: dato un albero ternario, associamo ai nodi più in alto le parole di codice dei simboli con probabilità maggiore.



Nell'immagine vediamo come prima inseriamo i simboli con probabilità 0.3 e 0.2, poi come terzo nodo mettiamo un "checkpoint" e iniziamo la procedura di nuovo da quest'ultimo nodo.

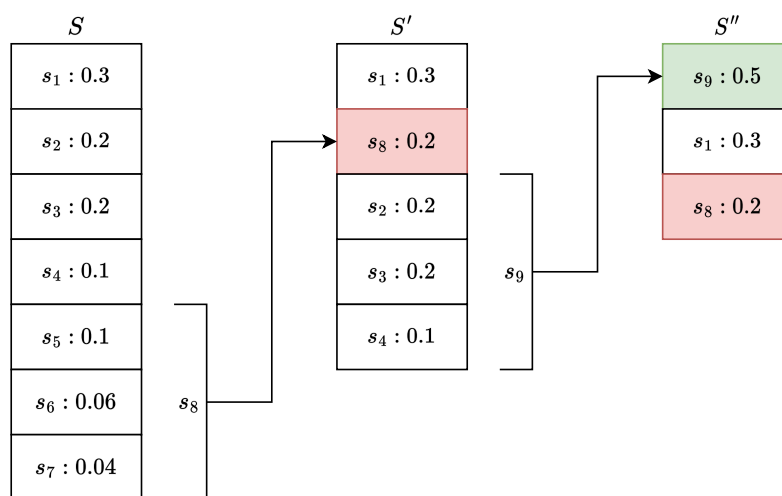
### 3.2.1. Definizione

Abbiamo in realtà un terzo approccio al problema precedente, ideato da **David Huffman** nel 1953, che lavora nel seguente modo:

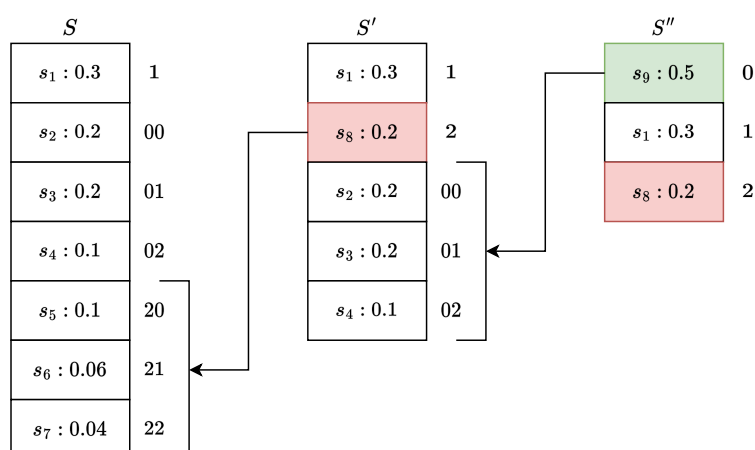
1. ordino le probabilità in ordine decrescente;
2. le ultime  $D$  probabilità sono sostituite dalla loro somma, e i simboli corrispondenti sono sostituiti da un simbolo “fantoccio”, creando una nuova sorgente “fantoccia”;
3. ripeto dal punto 1 fino a quando non si raggiungono  $t$  probabilità, con  $t \leq D$ ;
4. scrivo il codice di Huffman facendo un “rollback” alla sorgente iniziale.

Il codice così creato è detto **codice di Huffman**, ed è il *codice istantaneo ottimo* che possiamo costruire.

Supponiamo che la sorgente  $S$  emetta  $n = 7$  simboli  $s_1, \dots, s_7$  con probabilità  $P = \{0.3, 0.2, 0.2, 0.1, 0.1, 0.06, 0.04\}$ , vogliamo costruire un codice ternario di Huffman.

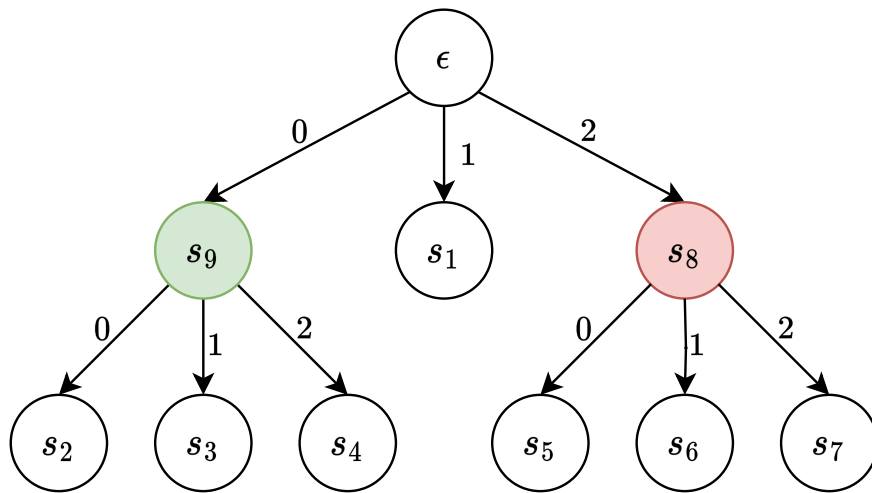


In questa prima fase andiamo a “compattare” le probabilità minori fino ad avere una situazione con esattamente  $D = 3$  simboli.



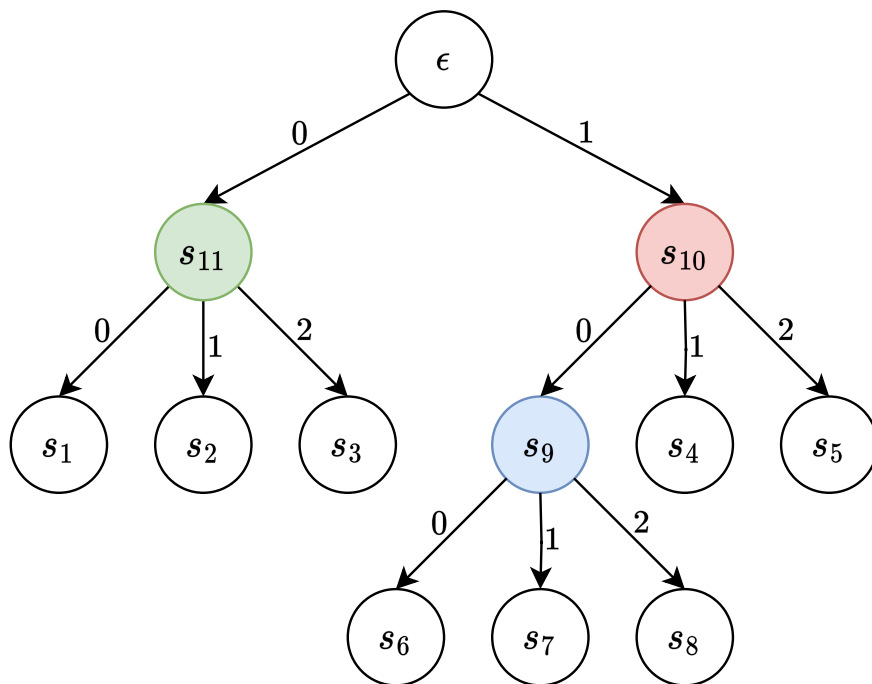
Nella seconda fase invece andiamo ad eseguire un “rollback” delle compressioni, sostituendo ad ogni nodo “compresso” le vecchie probabilità, andando quindi a costruire l’*albero* di codifica.





Supponiamo ora che la sorgente  $S$  emetta  $n = 8$  simboli  $s_1, \dots, s_7, s_8$  con probabilità  $P = \{0.3, 0.2, 0.2, 0.1, 0.1, 0.06, 0.02, 0.02\}$ , vogliamo costruire un codice ternario di Huffman.

Considerando che ad ogni iterazione perdiamo  $D = 3$  simboli e ne aggiungiamo uno, in totale perdiamo  $D - 1$  simboli. Considerando che l'algoritmo genera il codice istantaneo ottimo quando termina con esattamente  $D$  probabilità, in questo esempio alla fine delle iterazioni ci troveremmo con solo due simboli sorgente, e non tre, quindi la codifica non risulta ottimale.



Infatti, notiamo come alla radice perdiamo un ramo, andando ad aumentare di conseguenza l'altezza dell'albero.

La soluzione proposta da Huffman consiste nell'inserire un numero arbitrario di simboli "fantoccio" con probabilità nulla, così da permettere poi un'ottima "compressione" fino ad avere  $D$  simboli.

Ma quanti simboli nuovi dobbiamo inserire?

Supponiamo di partire da  $n$  simboli e rimuoviamo ogni volta  $D - 1$  simboli:

$$n \longrightarrow n - (D - 1) \longrightarrow n - 2 \cdot (D - 1) \longrightarrow \dots \longrightarrow n - t \cdot (D - 1).$$

Chiamiamo  $\blacksquare = n - t \cdot (D - 1)$ . Siamo arrivati ad avere  $\blacksquare$  elementi, con  $\blacksquare \leq D$ , ma noi vogliamo esattamente  $D$  elementi per avere un albero ben bilanciato e senza perdita di rami, quindi aggiungiamo a  $\blacksquare$  un numero  $k$  di elementi tali per cui  $\blacksquare + k = D$ . Supponiamo di eseguire ancora un passo dell'algoritmo, quindi da  $\blacksquare + k$  andiamo a togliere  $D - 1$  elementi, lasciando la sorgente con un solo elemento.

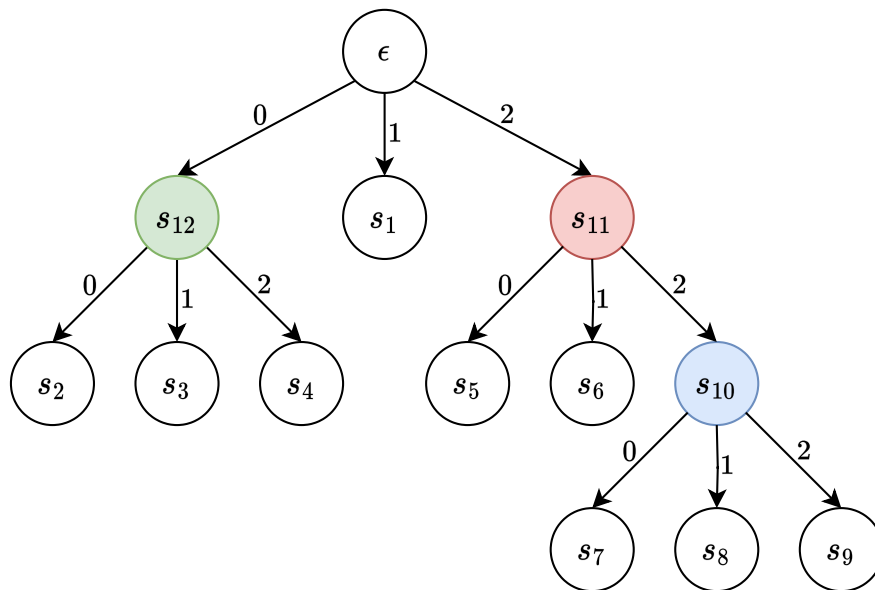
Cosa abbiamo ottenuto? Ricordando che  $\blacksquare = n - t \cdot (D - 1)$ , abbiamo fatto vedere che:

$$\begin{aligned}\blacksquare + k - (D - 1) &= 1 \\ n - t \cdot (D - 1) + k - (D - 1) &= 1 \\ n + k - (t + 1) \cdot (D - 1) &= 1.\end{aligned}$$

In poche parole, il numero  $n$  di simboli sorgente, aggiunto al numero  $k$  di simboli “fantoccio”, è congruo ad 1 modulo  $D - 1$ , ovvero

$$n + k \equiv 1 \pmod{D - 1}.$$

Ripetiamo l'esempio precedente, aggiungendo il simbolo  $s_9$  ad  $S$  con probabilità 0.



Con che ordine vado a inserire i simboli “compressi” dentro la lista delle probabilità? In modo *random*, quindi non è detto che il codice generato sia unico e ottimo.

## 4. Lezione 04

### 4.1. Entropia

#### 4.1.1. Introduzione

La scorsa lezione abbiamo introdotto il concetto di **entropia**, ovvero la quantità

$$H_{D(X)} = \sum_{i=1}^m p_i \log_D \left( \frac{1}{p_i} \right).$$

L'entropia dipende da  $D$ : se questo vale 2 l'entropia calcolata è quella **binaria** e non ha il pedice vicino alla  $H$ .

L'entropia fa solo riferimento alla distribuzione di probabilità, non c'entra niente con i simboli: come vediamo, nella definizione compare solo  $D$  assieme alle probabilità dei singoli simboli.

#### 4.1.2. Cambio di base

Sappiamo che con i logaritmi possiamo cambiare base. *E se volessi fare un cambio base nell'entropia?* Quello che sto facendo è cambiare l'entropia binaria in una che magari mi risulta più comoda.

Ricordando che

$$\log_b p = \frac{\log_c p}{\log_c b},$$

se fissiamo  $c = e$  otteniamo

$$\log_b p = \frac{\ln p}{\ln b} = \frac{\ln p}{\ln b} \cdot \frac{\ln a}{\ln a} = \frac{\ln p}{\ln a} \cdot \frac{\ln a}{\ln b} = \log_a p \cdot \log_b a.$$

In poche parole, *cambiare base* significa *cambiare base e portarsi dietro una costante*.

Modifichiamo quindi l'entropia, ottenendo

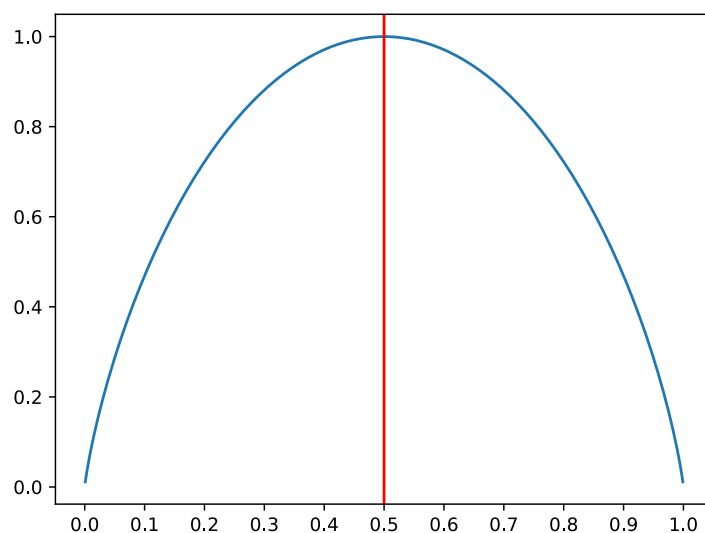
$$H_b(X) = \sum_{i=1}^m p_i \log_b \frac{1}{p_i} \longrightarrow H_a(X) = \sum_{i=1}^m p_i \log_a \frac{1}{p_i} \log_b a = \log_b a \cdot H_a(X).$$

#### 4.1.3. Esempio

Disegniamo l'entropia binaria usando  $\mathbb{X}$  variabile aleatoria bernoulliana.

Sia quindi  $P(\mathbb{X} = 1) = p$  e  $P(\mathbb{X} = 0) = 1 - p$ . Allora vale

$$H(X) = p \log_2 \frac{1}{p} + (1 - p) \log_2 \frac{1}{1 - p}.$$

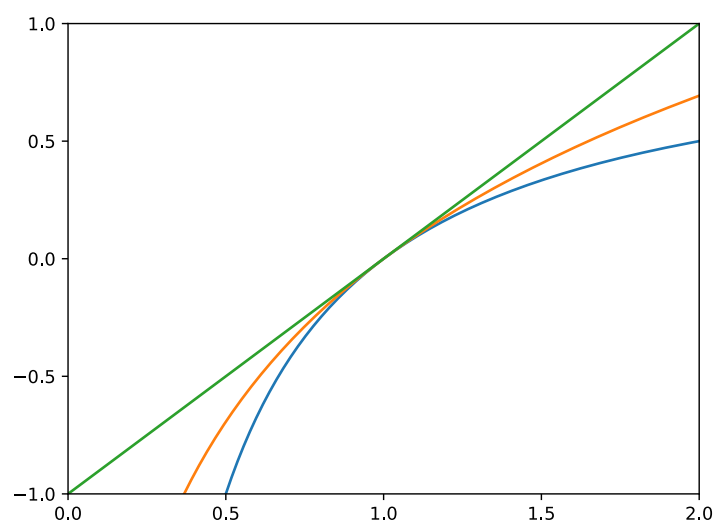


Notiamo come in  $p = 0/1$  l'entropia vale  $H(0/1) = 0$ , mentre è massima in  $p = \frac{1}{2}$  e vale  $H(\frac{1}{2}) = 1$ .

Infine, introduciamo la relazione

$$1 - \frac{1}{x} \leq \ln x \leq x - 1$$

che ci sarà utile dopo per alcune dimostrazioni.



#### 4.1.4. Proprietà

Diamo prima di tutto un upper bound all'entropia.

**Lemma**  $H_D(X) \leq \log_D m$ .

#### Dimostrazione

Dimostriamo che  $H_D(X) - \log_D m \leq 0$ .

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

Ricordiamo che  $\ln x \leq x - 1$ , quindi:

$$\begin{aligned}
\sum_{i=1}^m p_i \cdot \underbrace{\ln \frac{1}{p_i \cdot m}}_x \cdot \frac{1}{\ln D} &\leq \sum_{i=1}^m p_i \left( \frac{1}{p_i \cdot m} - 1 \right) \frac{1}{\ln D} \\
&\leq \frac{1}{\ln D} \sum_{i=1}^m \frac{1}{m} - p_i = \frac{1}{\ln D} \underbrace{\left[ \sum_{i=1}^m \frac{1}{m} - \sum_{i=1}^m p_i \right]}_0 = 0.
\end{aligned}$$

Se  $H_D(X) - \log_D m \leq 0$  allora

$$H_D(X) \leq \log_D m.$$

□

**Corollario** Se  $P(X = a_i) = \frac{1}{m} \quad \forall i = 1, \dots, m$  allora

$$H_D(X) = \log_D m.$$

### Dimostrazione

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

$$H_D(X) = \sum_{i=1}^m p_i \log_D \frac{1}{p_i} = \sum_{i=1}^m \frac{1}{m} \log_D m = \log_D m \underbrace{\sum_{i=1}^m \frac{1}{m}}_1 = \log_D m.$$

□

Introduciamo ora l'**entropia relativa**: indicata con  $\Delta(X \parallel Y)$  misura la distanza tra  $X$  e  $Y$ , ovvero la diversità tra  $X$  e  $Y$  in termini delle due distribuzioni di probabilità.

Definiamo quindi l'entropia relativa come

$$\Delta(X \parallel Y) = \sum_{s \in S} p_X(s) \log_D \frac{p_X(s)}{p_Y(s)}.$$

L'insieme  $S$  indica il **dominio** sul quale  $X$  e  $Y$  lavorano.

**Teorema** Per ogni coppia di variabili casuali  $X, Y$  definite sullo stesso dominio  $S$  vale la disuguaglianza

$$\Delta(X \parallel Y) \geq 0.$$

**Dimostrazione**

$$\begin{aligned} \Delta(X \parallel Y) &= \sum_{s \in S} p_X(s) \log_D \frac{p_X(s)}{p_Y(s)} = \sum_{s \in S} p_X(s) \ln \frac{p_X(s)}{p_Y(s)} \frac{1}{\ln D} = \\ &= \frac{1}{\ln D} \sum_{s \in S} p_X(s) \ln \underbrace{\frac{p_X(s)}{p_Y(s)}}_x = \text{uso } 1 - \frac{1}{x} \leq \ln x = \\ &\geq \frac{1}{\ln D} \sum_{s \in S} p_X(s) \left( 1 - \frac{p_Y(s)}{p_X(s)} \right) = \frac{1}{\ln D} \sum_{s \in S} p_X(s) - p_Y(s) \\ &\geq \frac{1}{\ln D} \left( \underbrace{\sum_{s \in S} p_X(s)}_1 - \underbrace{\sum_{s \in S} p_Y(s)}_1 \right) = 0. \end{aligned}$$

□

Infine, vediamo la relazione tra *valore atteso delle lunghezze del codice* e *entropia*.

**Teorema** Sia  $c : X \rightarrow \mathbb{D}^+$  un codice istantaneo  $D$ -ario per una sorgente  $\langle X, p \rangle$ , allora

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

**Dimostrazione**

Definisco  $\mathbb{Z} : X \rightarrow \mathbb{R}$  variabile casuale alla quale associamo una distribuzione di probabilità

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

Dimostriamo che  $\mathbb{E}[l_c] - H_D(X) \geq 0$ .

$$\begin{aligned}
\mathbb{E}[l_c] - H_D(X) &= \sum_{x \in X} p(x) l_c(x) - \sum_{x \in X} p(x) \log_D \frac{1}{p(x)} = \sum_{x \in X} p(x) \cdot \left( l_c(x) - \log_D \frac{1}{p(x)} \right) = \\
&= \sum_{x \in X} p(x) \cdot \left( \log_D D^{l_c(x)} - \log_D \frac{1}{p(x)} \right) = \\
&= \sum_{x \in X} p(x) \cdot (\log_D D^{l_c(x)} + \log_D p(x)) = \sum_{x \in X} p(x) \log_D (D^{l_c(x)} \cdot p(x)) = \\
&= \sum_{x \in X} p(x) \cdot \left( \log_D \left( \frac{p(x)}{D^{-l_c(x)}} \cdot 1 \right) \right) = \\
&= \sum_{x \in X} p(x) \cdot \left( \log_D \left( \frac{p(x)}{D^{-l_c(x)}} \cdot \frac{\sum_{x' \in X} D^{-l_c(x')}}{\sum_{x' \in X} D^{-l_c(x')}} \right) \right) = \\
&= \sum_{x \in X} p(x) \cdot \left( \log_D \left( p(x) \frac{\sum_{x' \in X} D^{-l_c(x')}}{D^{-l_c(x)}} \right) - \log_D \left( \sum_{x' \in X} D^{-l_c(x')} \right) \right) = \\
&= \sum_{x \in X} p(x) \cdot \left( \log_D \frac{p(x)}{q(x)} - \log_D \left( \sum_{x' \in X} D^{-l_c(x')} \right) \right) = \\
&= \sum_{x \in X} p(x) \log_D \frac{p(x)}{q(x)} - p(x) \log_D \left( \sum_{x' \in X} D^{-l_c(x')} \right) = \\
&= \sum_{x \in X} p(x) \log_D \frac{p(x)}{q(x)} - \sum_{x \in X} p(x) \log_D \left( \sum_{x' \in X} D^{-l_c(x')} \right) = \\
&= \underbrace{\Delta(X \parallel \mathbb{Z})}_{\geq 0} - \underbrace{\log_D \left( \sum_{x' \in X} D^{-l_c(x')} \right)}_{\substack{c \text{ istantaneo} \rightarrow \log_D(t \leq 1) \leq 0 \\ \geq 0}} \cdot \underbrace{\sum_{x \in X} p(x)}_{=1} \geq 0.
\end{aligned}$$

□

## 4.2. Sardinas-Patterson

L'**algoritmo di Sardinas-Patterson** è un algoritmo che permette di dimostrare se un codice è univocamente decodificabile.

### 4.2.1. Definizione

Dato  $A = \{x_1, \dots, x_n\}$  insieme dei simboli usati da un codice  $c$ , costruiamo l'insieme  $S_1 = A$ . Per stabilire se  $c$  è univocamente decodificabile generiamo una serie di insiemi  $S_i$  applicando le due regole seguenti:

- $\forall x \in S_1 \quad (\exists y \mid xy \in S_i \implies y \in S_{i+1})$ ;
- $\forall z \in S_i \quad (\exists y \mid zy \in S_1 \implies y \in S_{i+1})$ .

Quello che viene fatto è un controllo "incrociato":

1. parto da una parola  $x \in S_1$ ;
2. cerco se esiste una parola  $w \in S_i$  che abbia  $x$  come prefisso;
3. se esiste aggiungo  $y = w - x$  all'insieme  $S_{i+1}$ ;
4. faccio lo stesso invertendo  $S_1$  con  $S_i$ .

Fermiamo l'applicazione di queste regole quando:

- $S_i$  contiene un elemento di  $A$ : il codice  $c$  non è univocamente decodificabile;
- $S_i = \emptyset$ : il codice  $c$  è univocamente decodificabile.

#### 4.2.2. Esempi

Sia  $A = \{A, E, C, ABB, CED, BBEC\}$ , e sia  $S_1 = A$ .

Generiamo gli insiemi  $S_i$  con le due regole presentate.

- $S_2 = \{BB, ED\}$ ;
  - $BB$ :  $A \in S_1 \wedge ABB \in S_1$ ;
  - $ED$ :  $C \in S_1 \wedge CED \in S_1$ ;
- $S_3 = \{D, EC\}$ ;
  - $D$ :  $E \in S_1 \wedge ED \in S_2$ ;
  - $EC$ :  $BB \in S_2 \wedge BBEC \in S_1$ ;
- $S_4 = \{C\}$ ;
  - $C$ :  $E \in S_1 \wedge EC \in S_3$ .

Ci fermiamo perché  $C \in S_4$  e  $C \in S_1$ , quindi il codice non è univocamente decodificabile.



## 5. Lezione 05

### 5.1. Upper bound valore atteso

La scorsa lezione abbiamo mostrato che  $H_D(X) \leq \mathbb{E}[l_c]$ , ma *quanto sono vicini questi due valori?*

Se la distanza è alta vuol dire che il nostro codice è poco efficiente.

**Teorema** Dato  $c$  codice istantaneo di Shannon con lunghezze  $l_1, \dots, l_m$  tali che

$$l_i = \left\lceil \log_D \frac{1}{p_i} \right\rceil \quad \forall i \in \{1, \dots, m\}$$

allora

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

**Dimostrazione**

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

□

Questo teorema ci dice che il massimo spreco per simbolo rispetto all'ottimo è di un bit: sembra un ottimo risultato, ma se ho a disposizione tanti simboli si perdono molti bit.

### 5.2. Estrazione a blocchi

Dati  $c : X \rightarrow \mathbb{D}^+$  e  $C : X^+ \rightarrow D^+$  cerchiamo di dare una definizione dell'**estrazione a blocchi** di  $n$  con la funzione  $C_n : X^n \rightarrow D^+$ .

**Lemma**  $l_c(x_1, \dots, x_n) \geq l_{C_n}(x_1, \dots, x_n)$ .

**Dimostrazione**

$$\begin{aligned} l_C(x_1, \dots, x_n) &= \sum_{i=1}^n \left\lceil \log_D \frac{1}{p_i} \right\rceil \\ &\geq \left\lceil \sum_{i=1}^n \log_D \frac{1}{p_i} \right\rceil = \left\lceil \log_D \frac{1}{\prod_i p_i} \right\rceil = \left\lceil \log_D \frac{1}{p(x_1, \dots, x_n)} \right\rceil = l_{C_n}(x_1, \dots, x_n). \end{aligned}$$

□

In poche parole, la lunghezza di un messaggio trattato con i singoli simboli è non minore della lunghezza del messaggio stesso trattato a blocchi.

Infatti, definiamo

$$C_n : X^n \rightarrow D^+$$

come la funzione che estrae messaggi di  $n$  simboli. Prima avevamo l'estrazione di  $n$  simboli, ora l'estrazione di *un messaggio* di  $n$  simboli.

In questo modo paghiamo 1 bit ogni  $n$  simboli, ma abbiamo un modello sorgente più complesso.

Esiste una relazione tra  $H(x_1, \dots, x_n)$  e  $H(x)$ ?

Definiamo prima di tutto  $H(x_1, \dots, x_n)$  come

$$H(x_1, \dots, x_n) = \sum_{x_1, \dots, x_n} p_n(x_1, \dots, x_n) \log_2 \frac{1}{p_n(x_1, \dots, x_n)}.$$

Inoltre, sapendo che

$$p_n(x_1, \dots, x_n) = \prod_i p(x_i)$$

e che

$$\log_2 \frac{1}{\prod_{i=1}^n p(x_i)} = \log_2 \prod_{i=1}^n p(x_i)^{-1} = \sum_{i=1}^n \log_2 p(x_i)^{-1} = \sum_{i=1}^n \log_2 \frac{1}{p(x_i)},$$

possiamo affermare che

$$H(x_1, \dots, x_n) = \sum_{x_1} \dots \sum_{x_n} \prod_{i=1}^n p(x_i) \sum_{i=1}^n \log_2 \frac{1}{p(x_i)}.$$

Vediamo il caso  $n = 2$  per semplicità, poi diamo una versione generale.

$$\begin{aligned} H(x_1, \dots, x_n) &= \sum_{x_1} \sum_{x_2} \prod_{i=1}^2 p(x_i) \left( \log_2 \frac{1}{p(x_1)} + \log_2 \frac{1}{p(x_2)} \right) = \\ &= \sum_{x_1} \sum_{x_2} p(x_1)p(x_2) \log_2 \frac{1}{p(x_1)} + p(x_1)p(x_2) \log_2 \frac{1}{p(x_2)} = \\ &= \sum_{x_1} \sum_{x_2} p(x_1)p(x_2) \log_2 \frac{1}{p(x_1)} + \sum_{x_1} \sum_{x_2} p(x_1)p(x_2) \log_2 \frac{1}{p(x_2)} = \\ &= \sum_{x_1} p(x_1) \log_2 \frac{1}{p(x_1)} \sum_{x_2} p(x_2) + \sum_{x_1} p(x_1) \sum_{x_2} p(x_2) \log_2 \frac{1}{p(x_2)} = \\ &= H(x_1) + H(x_2). \end{aligned}$$

In generale possiamo affermare che

$$H(x_1, \dots, x_n) = nH(X).$$

### 5.3. Primo teorema di Shannon

**Teorema (Primo teorema di Shannon)** Sia  $C_n : X_n \rightarrow D^+$  un codice di Shannon  $D$ -ario a blocchi per la sorgente  $\langle X, p \rangle$ , ovvero

$$l_{C_n}(x_1, \dots, x_n) = \left\lceil \log_D \frac{1}{p_n(x_1, \dots, x_n)} \right\rceil.$$

Allora

$$\lim_{n \rightarrow \infty} \frac{1}{n} \mathbb{E}[l_{C_n}] = H_D(X).$$

### Dimostrazione

Sappiamo che

$$nH_D(X) = H_D(X_1, \dots, X_n) \leq \mathbb{E}[l_{C_n}] < H_D(X_1, \dots, X_n) + 1 = nH_D(X) + 1.$$

Dividiamo entrambi i membri per  $n$  e otteniamo

$$H_D(X) \leq \frac{1}{n} \mathbb{E}[l_{C_n}] < H_D(X) + \frac{1}{n}.$$

Se  $n \rightarrow \infty$  allora  $H_D(X) = H_D(X) + \frac{1}{n}$  e quindi

$$\frac{1}{n} \mathbb{E}[l_{C_n}] = H_D(X)$$

per il teorema degli sbirri. □

Il primo teorema di Shannon ci indica che il valore atteso si “schiaccia” sull’entropia col crescere della dimensioni dei blocchi: quindi, se facciamo crescere la dimensione del blocco paghiamo poco in termini di bit.

## 5.4. Approssimare i modelli sorgente

A volte non conosciamo a priori il modello  $\langle X, p \rangle$  che abbiamo a disposizione, quindi dobbiamo stimare quest’ultimo con un altro modello  $\langle Y, q \rangle$ .

Abbiamo a disposizione l’entropia relativa  $\Delta(X \parallel Y)$ : infatti, quest’ultima ci dice quanto sono distanti i due modelli.

**Teorema** Dato il modello sorgente  $\langle X, p \rangle$ , se  $c : X \rightarrow D^+$  è un codice di Shannon con  $l_c(x) = \left\lceil \log_D \frac{1}{q(x)} \right\rceil$ , dove  $q$  è una distribuzione di probabilità su  $X$ , allora

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

### Dimostrazione

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

□

## 6. Lezione 06

### 6.1. Algoritmo di Huffman

Avevamo già visto l'algoritmo di Huffman per la costruzione di un codice di Huffman, ma visto che non ce lo ricordiamo lo rivediamo.

Per costruire un codice di Huffman bisogna:

1. ordinare i simboli sorgente in base alla probabilità decrescente;
2. crea un modello fittizio in cui i  $D$  simboli meno probabili vengono raggruppati e sostituiti con un nuovo simbolo la cui probabilità è la somma delle probabilità dei simboli sostituiti;
3. ripeti dal punto 1 se la sorgente contiene più di  $D$  simboli.

Avevamo inoltre aggiunto un numero  $k$  di simboli a probabilità 0 per verificare che

$$m + k \bmod D - 1 \equiv 1,$$

con  $m$  numero di simboli sorgente.

### 6.2. Codici di Huffman

Presentiamo adesso un teorema permette di capire se il codice di Huffman, generato con l'algoritmo appena ripreso, sia buono o meno.

**Teorema** Dati una sorgente  $\langle X, p \rangle$  e  $D > 1$ , il codice  $D$ -ario  $c$  di Huffman minimizza  $\mathbb{E}[l_c]$  tra tutti i codici istantanei per la medesima sorgente.

Per dimostrare questo teorema dobbiamo prima enunciare un fatto.

**Lemma** Sia  $c'$  un codice  $D$ -ario di Huffman per la sorgente  $X' = \{x_1, \dots, x_{m-D+1}\}$  con probabilità  $p_1 \geq \dots \geq p_{m-D+1}$ . Sia  $X$  la sorgente ottenuta togliendo da  $X'$  il simbolo  $x_k$  e aggiungendo  $D$  nuovi simboli  $x_{m-D+2}, \dots, x_{m+1}$  con probabilità  $p_{m-D+2}, \dots, p_{m+1} < p_{m-D+1}$ . Inoltre, deve valere  $p_{m-D+2} + \dots + p_{m+1} = p_k$ . Allora vale

$$c(x) = \begin{cases} c'(x) & \text{se } x \neq x_k \\ i c'(x_k) & \text{se } k \in \{m-D+2, \dots, m+1\} \end{cases} \quad \forall i \in \{0, \dots, D-1\}$$

è un codice di Huffman per la sorgente.

Grazie a questo fatto dimostriamo il teorema precedente.

#### Dimostrazione

Dimostriamo per induzione su  $m$ .

Passo base:  $m = 2$

Huffman produce il codice  $c(x_1) = 0$  e  $c(x_2) = 1$  che è ottimo, qualunque sia la distribuzione di probabilità.

Passo induttivo:  $m > 2$

Supponiamo ora Huffman ottimo per  $k \leq m-1$  e dimostriamolo per  $m$ .

Fissiamo  $\langle X, p \rangle$  sorgente di  $m$  simboli. Sia  $X = \{\dots, u, \dots, v\}$ , con  $p(u), p(v)$  minime. Definiamo  $\langle X', p' \rangle$  con  $u, v \in X$  rimpiazzate da  $z \in X'$ . La funzione  $p'$  è tale che

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

Sia  $c'$  il codice di Huffman per  $\langle X', p' \rangle$ . Dato che  $|X'| = m - 1$  allora  $c'$  è ottimale per ipotesi induttiva.

Il codice  $c$  per  $X$  è definito come

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

Dimostriamo che il codice  $c$  è ottimo.

Esprimiamo il valore atteso

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

in termini di  $X'$  come

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

Per dimostrare l'ottimalità di  $c$  consideriamo un codice  $c_2$  per  $\langle X, p \rangle$  e verifichiamo che  $\mathbb{E}[l_c] \leq \mathbb{E}[l_{c_2}]$ .

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 massimi. Senza perdita di generalità assumiamo che  $r, s$  siano fratelli nell'albero di codifica di  $c_2$ . Infatti:

- se non fossero fratelli e avessero un altro fratello (chiamato  $f$  fratello di  $s$ ) scegliamo  $s$  e  $f$  al posto di  $s$  e  $r$ ;
- se non avessero fratelli possiamo sostituire le loro codifiche con quelle del padre fino a riportarci in una situazione in cui abbiano un fratello.

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

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

Abbiamo scambiato la codifica di  $r$  con quella di  $u$  e quella di  $s$  con quella di  $v$ .

Analizziamo  $\mathbb{E}[l_{\bar{c}_2}] - \mathbb{E}[l_{c_2}]$  per dimostrare che il primo valore è minore o uguale del secondo.

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

Sapendo che:

- $p(r) - p(u) \geq 0$  dato che  $u$  è minimo;
- $l_{c_2}(u) - l_{c_2}(r) \leq 0$ ;
- $p(s) - p(v) \geq 0$  dato che  $v$  è minimo;
- $l_{c_2}(v) - l_{c_2}(s) \leq 0$ .

Stiamo sommando due quantità negative, quindi

$$\mathbb{E}[l_{\bar{c}_2}] - \mathbb{E}[l_{c_2}] \leq 0 \implies \mathbb{E}[l_{\bar{c}_2}] \leq \mathbb{E}[l_{c_2}].$$

□

## 7. Lezione 07

### 7.1. Codici di Huffman

Non abbiamo finito la dimostrazione pookie!

#### Dimostrazione

Introduciamo ora il codice  $\bar{c}_{2'}$  fatto come segue:

$$\bar{c}_{2'} = \begin{cases} \bar{c}_2(x) & \text{se } x \neq z \\ \omega & \text{se } x = z \end{cases}.$$

Questo codice è definito sulla sorgente  $\langle X', p' \rangle$ . Inoltre, dopo aver scambiato  $r$  e  $s$  con  $u$  e  $v$ , questi ultimi sono fratelli, quindi:

- $\bar{c}_2(u) = \omega \cdot 0$ ;
- $\bar{c}_2(v) = \omega \cdot 1$ .

Ma allora

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

Mettendo insieme i due risultati otteniamo

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

Quindi

$$\mathbb{E}[l_c] \leq \mathbb{E}[l_{c_2}] \quad .$$

□

### 7.2. Disuguaglianza di Kraft-McMillan

Per cercare il codice ottimo ci siamo ristretti ai codici istantanei, ma così facendo rischiamo di lasciare fuori dei codici che potrebbero essere ottimi nonostante non siano istantanei.

Vedremo però che questi codici non esistono, dato che anche i codici univocamente decodificabili seguono la disuguaglianza di Kraft.

**Teorema** (*Disuguaglianza di Kraft-McMillan*) Siano  $l_1, \dots, l_m$  lunghezze positive. Allora queste ultime sono lunghezze delle parole di codice di un codice  $D$ -ario univocamente decodificabile per una sorgente di  $m$  simboli se e solo se

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

Per dimostrare questo teorema ci servirà l'estensione  $k$ -esima di un codice  $c$  definita come

$$C_k : X^k \rightarrow D^+.$$

### Dimostrazione

[ $\Leftarrow$ ] Questa implicazione è banale: infatti, se vale la disuguaglianza di Kraft, le lunghezze  $l_1, \dots, l_m$  sono le lunghezze di un codice istantaneo, che è anche univocamente decodificabile.

[ $\Rightarrow$ ] Vale  $\forall k \geq 1$  l'uguaglianza

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

Andiamo avanti dicendo che

$$\sum_{x_1} \dots \sum_{x_k} D^{-l_c(x_1)} \cdot \dots \cdot D^{-l_c(x_k)} = \sum_{x_k \in X^k} D^{-(l_c(x_1) + \dots + l_c(x_k))} = \sum_{x_k \in X^k} D^{-l_{C_k}(x_k)} .$$

In questo caso

$$l_{C_k}(x_k) = l_c(x_1) + \dots + l_c(x_k) .$$

Introduciamo l'insieme  $X_n^k$  come

$$\{x_k \in X^k \mid l_{C_k}(x_k) = n\} .$$

Andiamo ancora avanti con l'uguaglianza:

$$\sum_{x_k \in X^k} D^{-l_{C_k}(x_k)} = \sum_{n=1}^{kl_{\max}} \sum_{x_k \in X^k} D^{-l_{C_k}(x_k)} = \sum_{n=1}^{kl_{\max}} |X_n^k| D^{-n} .$$

Visto che  $c$  è univocamente decodificabile  $C$  è iniettiva, quindi  $|X_n^k| \leq |D^n|$ , e quindi

$$\sum_{n=1}^{kl_{\max}} |X_n^k| D^{-n} \leq \sum_{n=1}^{kl_{\max}} D^n D^{-n} \leq kl_{\max} ,$$

ma allora

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

Poniamo  $\sum_{x \in X} D^{-l_c(x)} = M$  e vediamo quanto vale questa quantità.

Visto che vogliamo  $M^x$  sotto  $xl_{\max}$ , allora  $M$  deve stare tra 0 e 1, quindi

$$\left( \sum_{x \in X} D^{-l_c(x)} \right)^k \leq 1 .$$

□



$$\begin{aligned} H(Y \mid X) &= \sum_{x \in X} p(x) H(Y \mid X = x) = \\ &= \sum_{x \in X} p(x) \left( \sum_{y \in Y} p(y \mid x) 2p(y \mid x) \right) = \\ &= \sum_{x \in X} \sum_{y \in Y} p(x, y) 2p(y \mid x). \end{aligned}$$

In questo caso:

- $p(x, y)$  è detta **probabilità congiunta** ed è la probabilità che avvengano  $x$  e  $y$ ;
- $p(x) = \sum_{y \in Y} p(x, y)$  è detta **probabilità marginale**;
- $p(y | x) = \frac{p(x, y)}{p(x)}$  è detta **probabilità condizionale**.

**Lemma** (*Chain Rule per l'entropia*) Vale la seguente uguaglianza:

$$H(X, Y) = H(X) + H(Y | X) = H(Y) + H(X | Y).$$

Vale inoltre, negli spazi condizionati, l'uguaglianza:

$$H(X, Y | Z) = H(X, Z) + H(Y | X, Z).$$

### 8.3. Esercizi su entropia

#### 8.3.1. Esercizio 01

Sia  $x \in X$  una variabile rappresentante l'estrazione di un numero tra 0 e 9, e sia  $Y$  definita come  $Y = X + 2 \bmod 10$ . Quanto vale  $H(Y|X)$ ?

RISPOSTA:

#### 8.3.2. Esercizio 02

Siano  $X = \{-1, 0, 1\}$  e  $Y = X^2$ . Quanto vale  $H(Y | X)$ ? E invece  $H(X | Y)$ ?

RISPOSTA:

#### 8.3.3. Esercizio 03

Dato un sistema SCR (sorgente-canale-ricevente), sia  $M$  una matrice che rappresenta il canale  $C$ .

$$M = \begin{pmatrix} & b_1 & b_2 & b_3 & b_4 & b_5 \\ a_1 & 0.3 & 0.1 & 0.3 & 0.1 & 0.1 \\ a_2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ a_3 & 0.3 & 0.3 & 0.1 & 0.1 & 0.2 \\ a_4 & 0.3 & 0.3 & 0.3 & 0.05 & 0.05 \end{pmatrix}.$$

Siano inoltre  $X = \{a_1, a_2, a_3, a_4\}$  e  $p = [0.2, 0.2, 0.2, 0.4]$  per  $S(?)$ .

Quanto vale  $H(R | S)$ ?

RISPOSTA:

#### 8.3.4. Esercizio 04

Calcolare  $H(R | S)$  come nell'esercizio precedente usando però

$$M = \begin{pmatrix} 0.2 & 0.2 & 0.3 & 0.2 & 0.1 \\ 0.2 & 0.5 & 0.1 & 0.1 & 0.1 \\ 0.6 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.3 & 0.1 & 0.1 & 0.1 & 0.4 \end{pmatrix}$$

e  $p = [0.2, 0.3, 0.1, 0.4]$ .

RISPOSTA:

#### 8.3.5. Esercizio 05

Posso ottenere lo stesso risultato dell'esercizio in un altro modo?

Ad esempio:

- usando la chain rule?
  - RISPOSTA:
- usando la chain rule, regola 1 modificata?
  - RISPOSTA: