Teoria dell'informazione e della trasmissione

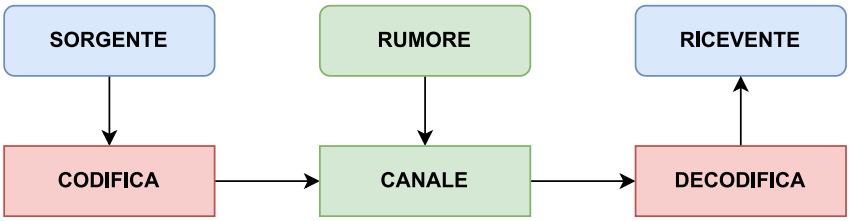
Indice

1. Lezione 01 [24/09]

2. Lezione 02 [01/10]

2.1. Introduzione storica

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 poteva accadere durante la trasmissione.

Nel 1948 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.

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 famoso e usatissimo **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.

2.2. Cosa faremo

Nel corso vedremo due operazioni fondamentali per spedire al meglio un messaggio sul canale:

- **compressione**: dobbiamo ottimizzare l’accesso al canale, ovvero se possiamo mandare meno bit ma questi mi danno le stesse informazioni dei bit totali ben venga;
- **aggiunta di ridondanza**: dobbiamo aggiungere dei bit per permettere il controllo, da parte del ricevente, dell’integrità del messaggio.

La compressione riguarda la **source coding**, e viene rappresentata nel **primo teorema di Shannon**, mentre la ridondanza riguarda la **channel coding**, e viene rappresentata nel **secondo teorema di Shannon**.

Voglio massimizzare l’informazione da spedire sul canale ma alla quale devo aggiungere ridondanza.

Quello che fa Shannon è modellare il canale secondo una matrice stocastica, che quindi permette di sapere in media cosa succede evitando tutti i casi precisi.

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

Questa matrice indica, per ogni carattere del nostro alfabeto, quale è la probabilità di spedire tale carattere e ottenere i vari caratteri presenti nell’alfabeto.

2.3. Esempio: informazione di un messaggio

Quando un messaggio contiene più informazioni di un altro?

Lancio n volte due monete, una truccata e una non truccata.

Quale delle due monete mi dà più informazioni? Sicuramente quella non truccata: il lancio della moneta «classica» è un evento randomico, non so mai cosa aspettarmi, mentre il lancio della moneta truccata è prevedibile, so già cosa succederà.

Possiamo quindi dire che un evento prevedibile porta pochissima informazione, mentre un evento imprevedibile porta tantissima informazione.

2.4. Esempio: codice ZIP

Uno dei codici di compressione più famosi è il codice **ZIP**.

Come funziona:

1. creo un dizionario, inizialmente vuoto, che contiene le coppie (stringa-sorgente,codifica), dove «codifica» è un numero;
2. usando un indice che scorre la stringa carattere per carattere, e partendo con numero di codifica uguale a 1, fino all’ultimo carattere eseguo iterativamente:
    1. parti da una stringa vuota che useremo come accumulatore;

2. aggiungi il carattere corrente all'accumulatore;
3. se l'accumulatore non è presente come chiave nel dizionario la aggiungo a quest'ultimo con codifica il numero corrente di codifica; riparto poi dal punto 2.1 con numero di codifica aumentato di 1;
4. se l'accumulatore è presente come chiave nel dizionario riparto dal punto 2.2.

Ad esempio, la stringa $AABACDABDCAABBA$ viene codificata con:

- $A \longrightarrow 1$;
- $A \longrightarrow AB \longrightarrow 2$;
- $A \longrightarrow AC \longrightarrow 3$;
- $D \longrightarrow 4$;
- $A \longrightarrow AB \longrightarrow ABD \longrightarrow 5$;
- $C \longrightarrow 6$;
- $A \longrightarrow AA \longrightarrow 7$;
- $B \longrightarrow 8$;
- $B \longrightarrow BA \longrightarrow 9$.

Prima avevo 15 caratteri, ora ne uso 9, letsgosky.

2.5. Richiami matematici

In questa parte abbiamo rivisto la definizione di monoide, gruppo, anello e campo, oltre alla definizione di generatore.

Abbiamo anche visto i **campi di Galois**, utilissimi per rendere il nostro calcolatore un campo algebrico, ovvero il campo $\mathbb{GF}(2^{64})$ dei polinomi di grado massimo 63 con coefficienti sul campo $\mathbb{Z}_2$.

3. Lezione 03 [04/10]

Dalla lezione scorsa abbiamo capito che dobbiamo fare due cose:

1. massimizzare l'informazione trasmessa ad ogni utilizzo del canale; il «ogni utilizzo» è importante perché ogni volta che utilizzo il canale devo essere top, adesso lo devo essere. Se devo mandare n informazioni lo posso fare in n accessi oppure in 1 accesso, in entrambi i casi massimizzo l'informazione trasmessa ma non uso sempre tutta la banda possibile;
2. minimizzare il numero di errori di trasmissione.

Shannon userà la tecnica del Divide Et Impera, ovvero risolverà le due task separatamente e poi unirà i due risultati parziali. Per puro culo, questa soluzione sarà ottima (*non sempre succede*).

Il punto 1 riguarda la sorgente, il punto 2 riguarda la codifica e il canale.

3.1. Codici

Prima di vedere un po' di teoria dei codici diamo alcune basi matematiche.

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 $D = \{0, ..., d-1\}$ insieme delle **parole di codice**.

Definiamo infine $c : X \longrightarrow D^+$ **funzione di codifica** che associa ad ogni carattere di X una parola di codice. Questa è una codifica in astratto: a prescindere da come è formato l'insieme X io uso le parole di codice che più mi piacciono (????). L'insieme D^+ è tale che

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

Voglio massimizzare la compressione: sia $l_c(x)$ la lunghezza della parola $x \in X$ nel codice c.

Quello che ho ora mi basta? NO: mi serve anche la **probabilità** $p(x)$ di estrarre un simbolo. Questo perché alle parole estratte molto 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 il codice morse.

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

Ora va bene? NON ANCORA: noi non vogliamo la probabilità di estrarre un simbolo perché noi lavoriamo con le parole, non con i simboli.

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 invece non è vera.

Il codice ZIP invece non assume l'indipendenza e lavora con la lingua che sta cercando di comprimere.

Dati il modello $\langle X, p \rangle$ e la base $d > 1$, dato 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?

Imponiamo che il codice sia un **codice non singolare**, ovvero che la funzione c sia iniettiva. 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 c è non singolare, come è C?

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

Andiamo a restringere l'insieme dei codici «buoni»: chiamiamo codici **univocamente decodificabili** (*UD*) i codici che hanno C non singolare.

Per dimostrare che un codice è UD esista l'algoritmo di Sardinas-Patterson, che lavora in tempo $O(mL)$, con $m = |X|$ e $L = \sum_{x \in X} l_c(x)$.

Ora però abbiamo un altro problema: gli UD 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.

Potremmo utilizzare 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.

Restringiamo per l'ultima volta, definendo i **codici istantanei** (*CI*). Questi codici 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.

Per capire se un codice è CI devo guardare i prefissi: nessuna parola di codice deve essere prefissa di un'altra. Questo controllo è molto più veloce rispetto al controllo che dovevo fare nei codici 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. Inoltre, il codice che abbiamo sotto mano è il migliore?

A noi gli UD andavano bene (*stream esclusi*), quanto pago per andare nei codici istantanei?

Teorema 3.1.1: Se un codice è istantaneo allora è anche univocamente decodificabile.

Dimostrazione 3.1.1: 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 (o viceversa), ma, quindi otteniamo che c non è istantaneo.

Ma allora il codice non è istantaneo. ■

4. Lezione 04 [08/10]

4.1. Disuguaglianza di Kraft

I codici UD ci vanno bene (stream escluso), mentre i CI sono perfetti ma perché ci danno di più ma ci fanno spendere più bit: quanti?

Noi stavamo minimizzando il valore atteso delle lunghezze delle parole di codice, ovvero

$$\min_{c \in \mathcal{C}} \quad \mathbb{E}[l_c] = \sum_{x \in X} l_c(x) p(x).$$

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

Date le lunghezze, l'unica cosa che posso dire è se un CI non esiste (quando sono sbagliate le lunghezze date), ma non posso dire che il codice che ho dato con quelle lunghezze è un CI, perché questo ci dice solo della sua esistenza, non se il codice che ho dato con quelle lunghezze è CI, per quello dovrei guardare i prefissi.

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.

Questa proprietà è detta **disuguaglianza di Kraft**.

Teorema 4.1.1 (Disuguaglianza di Kraft): Data una sorgente $X = \{x_1, ..., x_m\}$, data $d > 1$ base del codice e dati m interi $l_1, ..., l_m > 0$ che mi rappresentano le lunghezze dei simboli del codice, esiste un CI $c : X \longrightarrow D^+$ tale che

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

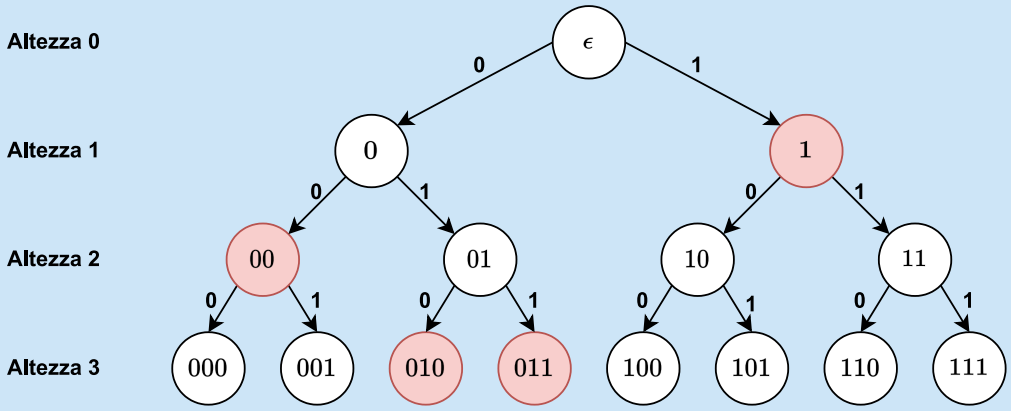
se e solo se

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

Dimostrazione 4.1.1:

($\Longrightarrow$) 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 che ogni nodo copre. Sono tutti alberi disgiunti, visto che non posso avere prefissi essendo c un CI. Il numero massimo di foglie è $d^{l_{\max}}$, noi non le potremmo coprire 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 di inserirsi nel suo sotto-albero;
- di inserire nodi che conterrebbero dei nodi già inseriti.

Una volta costruito l'albero vedo che 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. ■

Il nostro obiettivo rimane sempre e comunque 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, voglio che valga anche Kraft, così da avere un codice che sia istantaneo, quindi devono valere contemporaneamente

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

con $p_i = p(x_i) \quad \forall i = 1, ..., m$. Cosa possiamo dire?

Notiamo che

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

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

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

Posso quindi costruire un codice mettendo in relazione le lunghezze del codice con la probabilità di estrarle dalla sorgente.

Esempio 4.1.1: Dati

$$X = \{x_1, x_2, x_3, x_4\} \quad | \quad P = \left\{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}\right\} \quad | \quad d = 2$$

costruire un codice con la tecnica appena vista.

- $l_1 \geq \lceil \log_2(2) \rceil = 1 \longrightarrow 0$;
- $l_2 \geq \lceil \log_2(4) \rceil = 2 \longrightarrow 10$;
- $l_3 \geq \lceil \log_2(8) \rceil = 3 \longrightarrow 110$;
- $l_4 \geq \lceil \log_2(8) \rceil = 3 \longrightarrow 111$.

Questo codice ha valore atteso delle lunghezze

$$\mathbb{E}[l_c] = \frac{2}{4} + \frac{2}{4} + \frac{3}{4} = \frac{7}{4}.$$

Il valore atteso ora diventa

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

grazie alla nuova definizione delle lunghezze delle parole di codice.

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*. Il bro Fano era il dottorando di Shannon).

Inoltre, notiamo come il valore atteso delle lunghezze dipenda solo dalla distribuzione di probabilità dei simboli sorgente.

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.

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

- se ho una probabilità bassa di estrarre il simbolo x_i allora $\frac{1}{p_i}$ è grande e $\log_d\left(\frac{1}{p_i}\right)$ è grande;
- se ho una probabilità alta di estrarre il simbolo x_i allora $\frac{1}{p_i}$ è piccolo e $\log_d\left(\frac{1}{p_i}\right)$ è piccolo.

Purtroppo, i codici di Shannon-Fano non sono ottimali: una sorgente con due simboli a probabilità $p_1 = 0.1$ e $p_2 = 0.9$ darà un codice che non è ottimale.

Esempio 4.1.2: Siano

$$m = 4 \quad | \quad c : 1, 011, 01, 111.$$

Fai alcune considerazioni su questo codice.

Sicuramente non è CI: abbiamo dei prefissi che non vanno bene. Non è nemmeno UD: abbiamo una codifica ambigua per 111.

Esempio 4.1.3: Siano

$$m = 5 \quad | \quad c : 1, 001, 0000, 01, 0001.$$

Fai alcune considerazioni su questo codice.

Sicuramente è CI: non abbiamo prefissi, quindi è anche UD.

Posso vederlo come codice a virgola con il simbolo 1 che fa da separatore tra tutte le sequenze di 0.

Esempio 4.1.4: Siano

$$m = 5 \quad | \quad c : 000, 001, 01, 111, 110.$$

Fai alcune considerazioni su questo codice.

Sicuramente è CI: non abbiamo prefissi, quindi è anche UD.

Non è ottimale perché due foglie dell'albero non vengono coperte. Si potrebbe ridurre $\mathbb{E}[l_c]$ se scambiassimo 110 con 10 e 111 con 11.

Esempio 4.1.5: Siano

$$X = \{x_1, ...x_6\} \quad | \quad d = 2 \quad | \quad P = \left\{\frac{1}{15}, \frac{1}{3}, \frac{1}{6}, \frac{1}{9}, \frac{1}{5}, \frac{1}{29}\right\}.$$

Costruisci un codice di Shannon per queste lunghezze.

Le probabilità non sommano a 1.

Esempio 4.1.6: Siano

$$X = \{x_1, ...x_6\} \quad | \quad d = 2 \quad | \quad P = \left\{\frac{1}{12}, \frac{1}{3}, \frac{1}{5}, \frac{1}{3}, \frac{1}{72}, x\right\}.$$

Costruisci un codice di Shannon per queste lunghezze.

Le probabilità devono sommare a 1 quindi

$$p_6 = 1 - \left(\frac{1}{12} + \frac{1}{3} + \frac{1}{5} + \frac{1}{3} + \frac{1}{72}\right) = 1 - \frac{347}{360} = \frac{13}{360}.$$

- $l_1 \geq \lceil \log_2(12) \rceil = 4 \longrightarrow 1100$;
- $l_2 \geq \lceil \log_2(3) \rceil = 2 \longrightarrow 01$;
- $l_3 \geq \lceil \log_2(5) \rceil = 3 \longrightarrow 100$;
- $l_4 \geq \lceil \log_2(3) \rceil = 2 \longrightarrow 00$;
- $l_5 \geq \lceil \log_2(72) \rceil = 7 \longrightarrow 1111111$;
- $l_6 \geq \lceil \log_2(\approx 28) \rceil = 5 \longrightarrow 11010$.

5. Lezione 05 [11/10]

5.1. Entropia e tante dimostrazioni

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.

Cosa succede se vogliamo cambiare la base dell'entropia? 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 $\log_d(a)$.

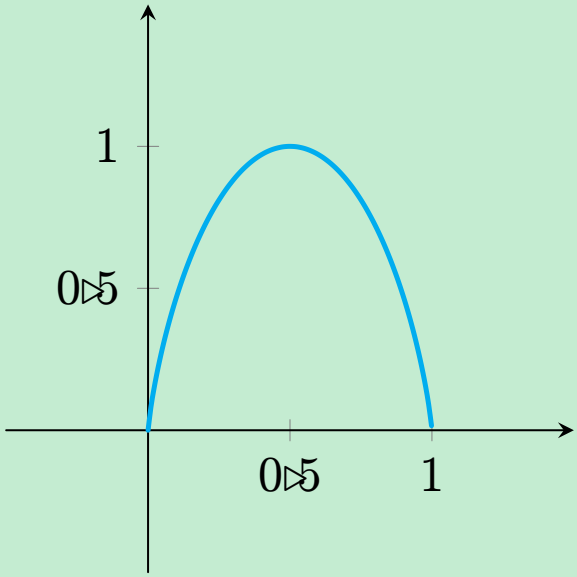
Esempio 5.1.1: Sia $\mathbb{X} : X \longrightarrow \{0, 1\}$ variabile aleatoria bernoulliana tale che

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

Calcoliamo l'entropia

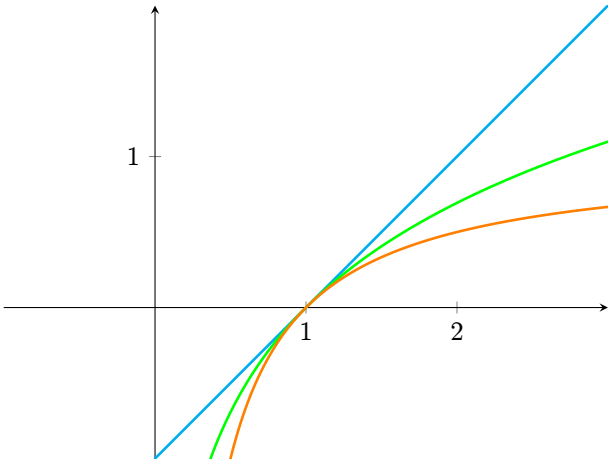
$$H(\mathbb{X}) = p \log_2\left(\frac{1}{p}\right) + (1 - p) \log_2\left(\frac{1}{1 - p}\right)$$

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 (*senza label lol non sono capace*) abbiamo che

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

Teorema 5.1.1: 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 5.1.1: 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

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

Ma allora

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

In particolare, se

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

allora

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

Nella scorsa lezione 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 l'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 $\mathbb{D}(\mathbb{X} \parallel \mathbb{W}) \neq \mathbb{D}(\mathbb{W} \parallel \mathbb{X})$ perché essa non è una distanza metrica, ma solo una distanza in termini di diversità.

Teorema 5.1.2 (Information inequality):

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

Dimostrazione 5.1.2: Come nella dimostrazione precedente, se volessimo cambiare la base del logaritmo avremmo un fattore moltiplicativo davanti alla sommatoria, che però non cambia la dimostrazione successiva. Noi manteniamo la base d in questa dimostrazione.

Valutiamo

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

Ma allora

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

■

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

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

Teorema 5.1.3: 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 5.1.3: 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 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.

5.2. Esercizi

Vediamo l'algoritmo di Sardinas-Patterson per verificare se un codice dato è UD.

Dato l'insieme S_1 contenente le parole del codice c, l'algoritmo esegue i seguenti passi:

1. si parte da $i = 1$;
2. sia $x \in S_1$, se esiste $xy \in S_i$ allora $y \in S_{i+1}$;
3. sia $z \in S_i$, se esiste $zy \in S_1$ allora $z \in S_{i+1}$;
4. osserviamo S_{i+1}:
   - se $S_{i+1} = \varnothing$ allora c è UD;
   - se $S_{i+1} \cap S_1 \neq \varnothing$ allora c non è UD;
   - se S_{i+1} no rientra nei due casi precedenti ritornare al punto 2 con $i = i + 1$.

Esempio 5.2.1: Sia $S_1 = \{A, E, C, ABB, CED, BBEC\}$, il codice c con le parole di codice contenute in S_1 è UD?

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

Ma $S_1 \cap S_4 \neq \varnothing$ quindi c non è UD.

6. Lezione 06 [15/10]

6.1. Molti teoremi

Teorema 6.1.1: 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 $l_i = l_c(x_i) \quad \forall i = 1, ...m$

costruite con

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

.

Vale

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

Dimostrazione 6.1.1: 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 lo uniamo al bound trovato nelle scorse lezioni otteniamo

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

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 si accumula.

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

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

Shannon allora propone una soluzione, che come vedremo non funzionerà nel caso reale.

Cosa fa Shannon? Vediamo un esempio.

Esempio 6.1.1: 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 6.1.2: Sia $X = \{0, ..., 9\}$, 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.

Vedremo dopo che se la dimensione del blocco cresce all'infinito il codice di Shannon che sto costruendo è ottimale, altrimenti stiamo sprecando delle informazioni.

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 $(x_1, ..., x_n)$, e contiene n simboli estratti tramite estrazioni indipendenti e identicamente distribuite, ovvero

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

Questa quantità verrà indicata con

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

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 (*n 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 6.1.3: 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 6.1.2 (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 6.1.2: La dimostrazione è banale:

$$H_d(\mathbb{X}_1, ..., \mathbb{X}_n) \leq \mathbb{E}\left[l_{C_n}\right] < H_d(\mathbb{X}_1, ..., \mathbb{X}_n) + 1$$

$$nH_d(\mathbb{X}) \leq \mathbb{E}\left[l_{C_n}\right] \leq nH_d(\mathbb{X}) + 1$$

$$\frac{1}{n}(nH_d(\mathbb{X})) \leq \frac{1}{n}\mathbb{E}\left[l_{C_n}\right] \leq \frac{1}{n}(nH_d(\mathbb{X}) + 1)$$

$$H_d(\mathbb{X}) \leq \frac{1}{n}\mathbb{E}\left[l_{C_n}\right] \leq H_d(\mathbb{X}) + \frac{1}{n}.$$

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 $\mathbb{E}\left[l_{C_n}\right]$.

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

Teorema 6.1.3: Data una sorgente $\langle X, p \rangle$, se $c : X \longrightarrow D^+$ è un codice di Shannon avente lunghezze

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

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

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

Dimostrazione 6.1.3: Banale anche questo:

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

Odio il quadratino. ■

6.2. Esercizi

Esempio 6.2.1: Sia $S_1 = \{A, E, C, ABB, CED, BBEC\}$ l'insieme dell'esercizio della scorsa lezione, eseguire Sardinas-Patterson ma non fermarsi quando si trova un elemento che apparteneva a S_1.

- $S_1 = \{A, E, C, ABB, CED, BBEC\}$;
- $S_2 = \{BB, ED\}$;
- $S_3 = \{D\} \cup \{EC\} = \{D, EC\}$;
- $S_4 = \{C\} \cup \{\varnothing\} = \{C\}$;
- $S_5 = \{ED\}$;
- $S_6 = \{D\}$;
- $S_7 = \varnothing$.

L'algoritmo accetterebbe c anche se non è UD.

Esempio 6.2.2: Sia $S_1 = \{0, 01, 10\}$, il codice c con le parole di codice contenute in S_1 è UD?

- $S_1 = \{0, 01, 10\}$;
- $S_2 = \{1\}$;
- $S_3 = \{0\}$.

Ma $S_1 \cap S_3 \neq \varnothing$ quindi c non è UD.

Se non ci fermiamo quando troviamo un elemento di S_1 otteniamo:

- $S_4 = \{1\}$.

Ma $S_4 = S_2$, quindi abbiamo trovato un insieme che abbiamo già visitato e quindi il codice c non viene accettato di nuovo.

Va aggiunto infatti un altro caso di terminazione a Sardinas-Patterson, ovvero l'algoritmo termina non accettando il codice c come UD quando trova un insieme che è già stato incontrato precedentemente.

Esempio 6.2.3: Sia

$$P = \left\{\frac{1}{3}, \frac{1}{4}, \frac{1}{6}, \frac{1}{8}, \frac{1}{9}, x\right\}.$$

Calcolare le lunghezze di codice per un codice di Shannon c che abbia 6 simboli estratti le probabilità di P.

Calcoliamo

$$x = 1 - \left(\frac{1}{3} + \frac{1}{4} + \frac{1}{6} + \frac{1}{8} + \frac{1}{9}\right) = 1 - \frac{24 + 18 + 12 + 9 + 8}{72} = 1 - \frac{71}{72} = \frac{1}{72}.$$

Calcoliamo le lunghezze come:

- $l_1 = \lceil \log_2 3 \rceil \longrightarrow 2$;
- $l_2 = \lceil \log_2 4 \rceil \longrightarrow 2$;
- $l_3 = \lceil \log_2 6 \rceil \longrightarrow 3$;
- $l_4 = \lceil \log_2 8 \rceil \longrightarrow 3$;
- $l_5 = \lceil \log_2 9 \rceil \longrightarrow 4$;
- $l_6 = \lceil \log_2 72 \rceil \longrightarrow 7$.

L'albero di codifica non copre tutte le foglie: infatti, questo codice non è ottimale perché ci sono dei rami dell'albero che possiamo potare.

7. Lezione 07 [19/10]

7.1. Codice di Huffman

Il codice di Shannon era un codice interessante, aveva dei buoni bound ma non era il CI ottimale. Inoltre, era 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 esegue i seguenti passi:

- ordina le probabilità p_i in maniera decrescente;
- 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;
- se la nuova sorgente ha più di d simboli torna al punto 1.

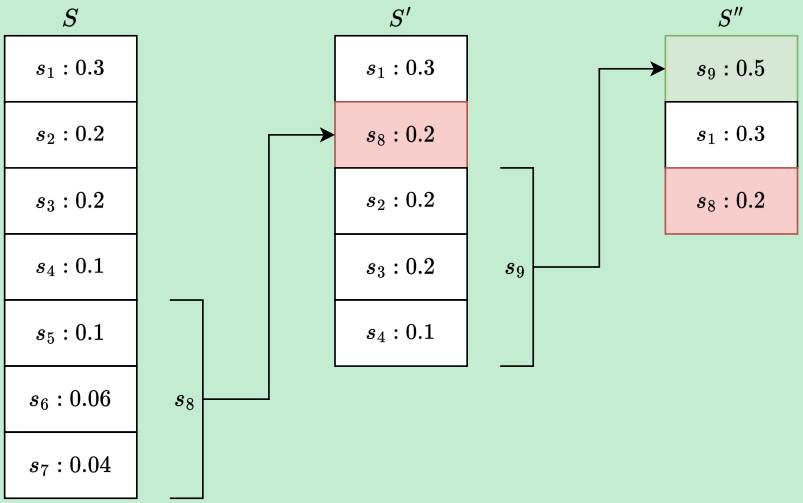
Come funziona nella realtà? Vediamolo con un esempio.

Esempio 7.1.1: Data la sorgente $\langle S, p \rangle$ con:

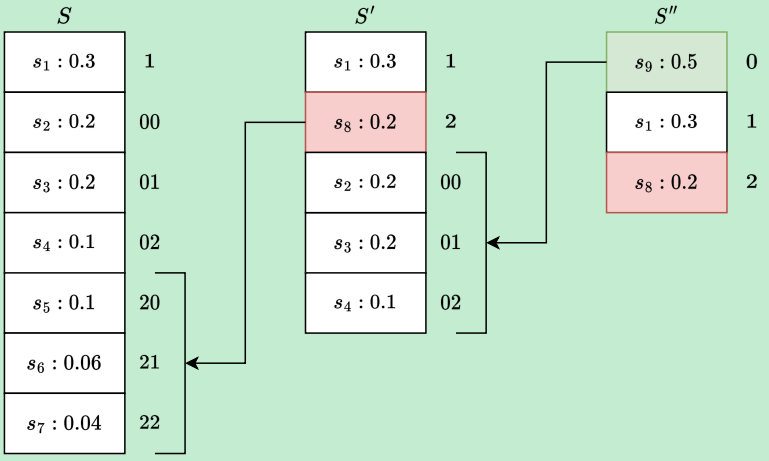
- $S = \{s_1, ..., s_7\}$ e
- $P = \{0.3, 0.2, 0.2, 0.1, 0.1, 0.06, 0.04\}$,

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:

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

Lemma 7.1.1: 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

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

è un codice di Huffman per la sorgente X.

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

Teorema 7.1.1: 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 7.1.1: Dimostriamo per induzione su m.

Il passo base è $m = 2$, quindi abbiamo due simboli x_1, 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

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

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

8. Lezione 08 [22/10]

8.1. Disuguaglianza di Kraft-McMillan

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

Riprendiamo alcuni concetti utili per il prossimo teorema.

Che codici abbiamo visto fin'ora? Ne abbiamo visti tre:

- $c : X \longrightarrow D^+$ codifica iniziale;
- $C : X^+ \longrightarrow D^+$ estensione di c;
- $C_n : X^n \longrightarrow D^+$ codice a blocchi.

A noi ora non serve il codice a blocchi ma solo l'estensione, ma modifichiamola leggermente: sia $C_k : X^k \longrightarrow D^+$ l'estensione k-esima del codice c, con $k \geq 1$, tale che

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

Il codice c deve essere sicuramente non singolare, ma anche C_k lo deve essere, quindi il codice è UD.

Le lunghezze di C_k sono

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

Per convenzione sia

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

Ma allora

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

Siamo pronti per dimostrare il seguente teorema.

Teorema 8.1.1 (Disuguaglianza di Kraft-McMillan): Data una sorgente $X = \{x_1, ..., x_m\}$, data $d > 1$ base del codice e dati m interi $l_1, ..., l_m > 0$ che mi rappresentano le lunghezze dei simboli del codice, esiste un codice UD $c : X \longrightarrow D^+$ tale che

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

se e solo se

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

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

Dimostrazione 8.1.1: Dimostriamo la doppia implicazione.

$\{\Longleftarrow\}$

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

$\{\Longrightarrow\}$

Partiamo dal valore

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

e vediamo che valori assume la quantità

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

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

Esempio 8.1.1:

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

Visto questo esempio, possiamo dire che

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

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

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

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

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

Nell'insieme $X_{kl_{\max}}^k$ abbiamo le parole di lunghezza massima.

Riscriviamo la sommatoria come

$$\begin{aligned} \sum_{x_1 \ldots x_k \in X^k} d^{-l_{C_k}(x_1, \ldots, x_k)} &= \sum_{n=1}^{kl_{\max}} \sum_{(x_1, \ldots, x_k) \in X_n^k} d^{-l_{C_k}(x_1, \ldots, x_k)} = \\ &= \sum_{n=1}^{kl_{\max}} \sum_{(x_1, \ldots, x_k) \in X_n^k} d^{-n} = \end{aligned}$$

8.2. Esercizi

Esempio 8.2.1: Il signor Giovanni Rossi ha

$$X = \{x_1, x_2, x_3, x_4, x_5\} \quad | \quad D = \{000, 001, 01, 110, 111\}.$$

Giovanni ha costruito il codice migliore usando una certa distribuzione P_i ma il bro non si ricorda quale ha utilizzato tra:

- $P_1 = \{0.2, 0.2, 0.2, 0.2, 0.2\}$;
- $P_2 = \{0.4, 0.2, 0.2, 0.1, 0.1\}$;
- $P_3 = \{0.1, 0.1, 0.2, 0.4, 0.2\}$.

Quale di queste probabilità è stata utilizzata?

Calcoliamo i valori attesi:

$$\mathbb{E}[l_c \mid P_1] = \frac{3}{5} + \frac{3}{5} + \frac{2}{5} + \frac{3}{5} + \frac{3}{5} = \frac{14}{5}$$
$$\mathbb{E}[l_c \mid P_2] = \frac{12}{10} + \frac{6}{10} + \frac{4}{10} + \frac{3}{10} + \frac{3}{10} = \frac{14}{5}$$
$$\mathbb{E}[l_c \mid P_3] = \frac{3}{10} + \frac{3}{10} + \frac{4}{10} + \frac{12}{10} + \frac{6}{10} = \frac{14}{5}$$

Sono tutte uguali. Soluzione alternativa: con Huffman dovevamo vedere se le lunghezze che uscivano erano uguali a quelle date.

Esempio 8.2.2: Date

$$X = \{a_1, ..., a_8, a_9, ..., a_{12}\} \quad | \quad P = \left\{ \underbrace{0.1, ..., 0.1}_{1,...,8}, \underbrace{0.05, ..., 0.05}_{9,...,12} \right\}.$$

Scrivere un CI con $d = 5$.

Applicando Huffman si ottiene:

- $a_1 = 2$;
- $a_2 = 3$;
- $a_3 = 00$;
- $a_4 = 01$;
- $a_5 = 02$;
- $a_6 = 03$;
- $a_7 = 04$;
- $a_8 = 10$;
- $a_9 = 11$;
- $a_{10} = 12$;
- $a_{11} = 13$;
- $a_{12} = 14$.

Posso fare meglio perché non ho un numero giusto di simboli, 5 foglie non vengono coperte. Soluzione alternativa: creare l'albero di codifica cercando di coprire tutte le foglie.

Riguardo l'ultimo esercizio, come faccio se devo usare per forza Huffman? Il problema dell'ultimo esercizio è nel numero di simboli, che non sono nel numero 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 $\blacksquare = m-t(d-1)$. Siamo arrivati ad avere $\blacksquare$ elementi, con $\blacksquare \leq d$.

Noi vorremmo avere esattamente d elementi per avere un albero ben bilanciato e senza perdita di rami, quindi aggiungiamo a $\blacksquare$ un numero k di **simboli dummy** tali per cui $\blacksquare + k = d$. 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 $\blacksquare + k$ andiamo a togliere $d-1$ elementi, lasciando la sorgente con un solo elemento.

Cosa abbiamo ottenuto? Ricordando che $\blacksquare = m-t(d-1)$, 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

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

Esempio 8.2.3: Rifare l'esercizio precedente sapendo che possiamo utilizzare i simboli dummy.

Applicando Huffman si ottiene:

- $a_1 = 2$;
- $a_2 = 3$;
- $a_3 = 4$;
- $a_4 = 00$;
- $a_5 = 01$;
- $a_6 = 02$;
- $a_7 = 03$;
- $a_8 = 04$;
- $a_9 = 10$;
- $a_{10} = 11$;
- $a_{11} = 12$;
- $a_{12} = 13$;
- $a_{13} = 14$ (*questo è dummy*).

Ora l'albero di codifica non ha nessuna foglia scoperta.

9. Lezione 09 [25/10]

9.1. Parenti dell'entropia

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

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

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

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

Sapendo che

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

e che la *probabilità marginale* (servirà dopo) è

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

allora

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

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

Esempio 9.1.1: Sia $\mathbb{X}$ una variabile e sia $\mathbb{Y} = \mathbb{X} + 2$ un'altra variabile casuale. Quanto vale $H(\mathbb{Y} \mid \mathbb{X})$?

Data la dipendenza di $\mathbb{X}$ da $\mathbb{Y}$, allora $H(\mathbb{Y} \mid \mathbb{X}) = 0$.

Esempio 9.1.2: Sia $\mathbb{X}$ una variabile e sia $\mathbb{Y} = \mathbb{X}^2$ un'altra variabile casuale. Entrambe pescano dall'insieme $\{-1, 0, 1\}$. Quanto vale $H(\mathbb{Y} \mid \mathbb{X})$?

Data la dipendenza di $\mathbb{X}$ da $\mathbb{Y}$, allora $H(\mathbb{Y} \mid \mathbb{X}) = 0$.

Quanto vale invece $H(\mathbb{X} \mid \mathbb{Y})$?

Sicuramente $H(\mathbb{X} \mid \mathbb{Y}) > 0$ perché il valore 1 estratto da $\mathbb{Y}$ può essere ottenuto sia da 1 che da -1.

Teorema 9.1.1 (Chain rule per l'entropia):

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

Dimostrazione 9.1.1: Verifichiamo che

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

Odio il quadratino. ■

Questa regola vale anche negli spazi condizionati, ovvero

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

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

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

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

Lemma 9.1.1:

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

Dimostrazione 9.1.2: Valutiamo

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

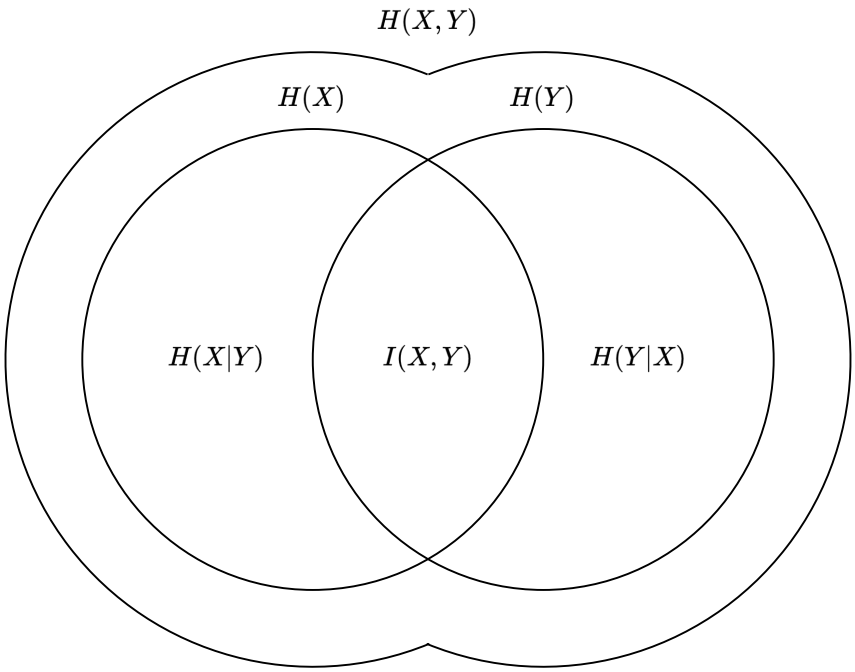
Quadratino di merda. ■

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

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

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

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



Teorema 9.1.2 (Data Processing Inequality): Siano $\mathbb{X}, \mathbb{Y}, \mathbb{W}$ variabili aleatorie con codominio finito tali che $p(x,y,w)$ soddisfa $p(x,w \mid y) = p(x \mid y)p(w \mid y) \quad \forall x,y,w$ (ovvero x,w indipendenti dalla y condizionante) allora

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

Dimostrazione 9.1.3: Valutiamo

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

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

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

ma allora

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

■

A cosa serve questo teorema? Abbiamo

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

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